ibUMAP: Coherent and Scalable Field Evaluation for UMAP Optimization

Anonymous authors
Paper under double-blind review

Abstract

UMAP achieves scalable layout optimization through stochastic negative sampling. However, this stochasticity can lead to unstable embeddings across reruns and downstream reuse, as the estimated repulsive forces depend on the ordering of sampling events. We present ibUMAP, a coherent field-based alternative that evaluates attraction and repulsion from a shared embedding snapshot and applies them synchronously. Its degree-weighted repulsive field is motivated by the conditional expectation of negative sampling for a fixed embedding and represented by three scalar moments, which are evaluated efficiently on CPUs and GPUs using an interpolation-based FFT scheme. This formulation avoids explicit all-pairs computations while inducing optimization dynamics that differ from those of standard online UMAP. Controlled experiments show that synchrony and kernel capping alter the local–global fidelity trade-off, whereas FFT evaluation produces small average changes in final quality. End-to-end benchmarks show median speedups of $3.29\times$ unseeded and $5.79\times$ seeded over umap-learn on CPU, and $1.44\times$ over cuML on million-scale datasets under unseeded GPU execution. These gains accompany greater run-to-run stability and measurable fidelity trade-offs.

1 Introduction

Uniform Manifold Approximation and Projection (UMAP) is a widely used neighbor-embedding technique whose practical optimization scheme combines sampled attraction over the fuzzy graph with stochastic negative sampling in the embedding space (McInnes et al., 2020). This online formulation circumvents the need to evaluate an explicit dense non-edge term, but its repeated random sampling and in-place pairwise updates make the resulting optimization trajectory sensitive to the ordering of events. Such sensitivity becomes particularly relevant when embeddings are repeatedly computed at scale or serve as intermediate representations in subsequent analysis pipelines.

The expectation underlying UMAP's negative-sampling optimizer offers a natural basis for an alternative treatment of its repulsive interactions (Damrich & Hamprecht, 2021). Drawing on earlier N-body formulations of nonlinear embedding gradients (Vladymyrov & Carreira-Perpiñán, 2014) and the interpolation-based FFT approach developed for FIt-SNE (Linderman et al., 2019), we express degree-weighted repulsion as a continuous field that can be evaluated efficiently without explicit enumeration of all point pairs. Rather than accelerating UMAP's stochastic sampling scheme, as in existing GPU implementations (Nolet et al., 2021), this formulation explores an alternative computational regime with distinct optimization dynamics. ibUMAP retains UMAP's graph construction and initialization algorithms, modifying only the layout-optimization stage. Figure 1 contrasts the update mechanisms in panel (a) and illustrates the three-moment field evaluation in panel (b).

Replacing online sampled interactions with a synchronous field changes the optimizer as well as its computational strategy. Conclusions derived for pairwise UMAP updates (Islam & Fleischer, 2026) do not necessarily extend to synchronous force aggregation. Because informative initialization can itself affect apparent global structure (Kobak & Linderman, 2021), we hold the graph and initialization fixed when studying optimizer effects. We therefore investigate two related questions: how to efficiently evaluate the degree-weighted repulsive field for a fixed embedding, and how synchronous updates, repulsion formulation, ibFFT field evaluation, and safeguards affect final embedding quality under controlled graph construction and initialization. These controlled comparisons complement end-to-end benchmarks of speed, fidelity, and repeatability.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

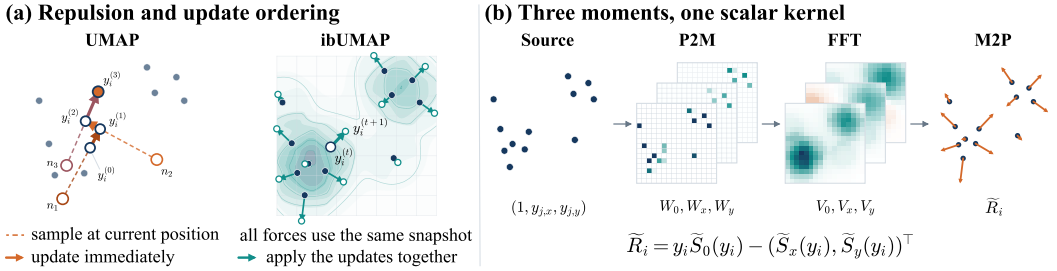


Figure 1: **ibUMAP: from sampled in-place updates to coherent field evaluation.** **(a)** From shared initial coordinates, UMAP negative-sampling events update the head immediately, whereas ibUMAP evaluates a radial repulsive field at one snapshot and applies updates together. **(b)** Particle-to-mesh (P2M) deposition, FFT convolution with a shared capped scalar kernel, and mesh-to-particle (M2P) interpolation reconstruct $\widetilde{R}_i$ from three scalar moments.

Contributions. *Formulation and implementation.* We formalize the frozen-state, degree-weighted repulsive operator under explicit negative-sampling and update conventions and derive a UMAP-specific three-moment decomposition, enabling synchronous two-dimensional FFT-based optimization on CPUs and GPUs with explicit complexity and repeatability conditions. *Mechanism decomposition.* We compare different optimizer mechanisms in a fixed-graph, fixed-initialization framework, finding that synchrony and kernel capping alter the local–global fidelity trade-off, whereas ibFFT evaluation has small average effects on final embedding quality. *Performance and repeatability.* We evaluate end-to-end speed, fidelity, and repeatability, finding median CPU speedups of $3.29\times$ unseeded and $5.79\times$ seeded over umap-learn, with additional unseeded GPU gains over cuML on million-scale datasets. We quantify the accompanying fidelity trade-offs and fixed-seed repeatability costs. In a BRAQUE-derived workflow, we trace embedding variation to HDBSCAN assignment changes and demonstrate ibUMAP's lower CPU cost for repeatable reuse (Dall'Olio et al., 2023).

2 Background and Related Work

UMAP's optimizer is not its objective. UMAP belongs to the neighbor-embedding lineage that includes t-SNE (van der Maaten & Hinton, 2008), but it builds and optimizes a sparse fuzzy neighborhood graph rather than a normalized dense input affinity matrix (McInnes et al., 2020). The fuzzy cross-entropy used to motivate UMAP is not the object that the implementation optimizes: practical UMAP schedules positive edges, draws negative vertices, and applies each interaction immediately. Later events therefore observe modified coordinates, making the trajectory sensitive to event ordering.

Negative sampling and its expectation. UMAP avoids enumerating the dense non-edge term of that cross-entropy by sampling negative vertices, and analyses of this estimator motivate degree-dependent repulsive weighting (Damrich & Hamprecht, 2021). Related work in large-scale graph layout reaches a complementary result from the sampling side: SNAP-tFDP uses edge-centric negative sampling whose expected repulsive force recovers a degree-weighted full-force formulation, while retaining stochastic in-place updates (Chen et al., 2026). Changing the effective balance between attraction and repulsion moves a method along a spectrum between more continuous and more clustered representations (Böhm et al., 2022), and negative sampling also links UMAP to contrastive learning (Damrich et al., 2023). Attraction shape further controls contraction, annealing, cluster formation, and sensitivity to initialization in UMAP-style pairwise dynamics (Islam & Fleischer, 2026). These analyses motivate the controlled comparisons in Section 4, which measure how optimizer mechanisms affect final embedding quality under a fixed graph and initialization.

Scalable N-body and field evaluation. Dense nonlinear-embedding gradients have been expressed as N-body sums and accelerated with fast multipole methods (Vladymyrov & Carreira-Perpiñán, 2014). GPGPU recast t-SNE repulsion as scalar and vector fields over the two-dimensional embedding domain and approximated them on GPUs using texture splatting (Pezzotti et al., 2020),

while FIt-SNE established the particle-to-mesh, FFT convolution, and mesh-to-particle pattern for dense neighbor-embedding forces (Linderman et al., 2019); the Fast Kernel Transform provides a kernel-general, adaptive, accuracy-controlled alternative to a regular-grid FFT scheme (Ryan et al., 2022), and FFT/GPU field evaluation has also been applied to force-directed graph layouts (Zhong et al., 2024). Unlike t-SNE field formulations, ibUMAP targets the degree-weighted repulsive operator induced by UMAP-style optimization semantics and studies its use inside a synchronous optimizer; cuML instead accelerates the sampled UMAP pipeline on GPUs (Nolet et al., 2021) and is our primary accelerated baseline. FastUMAP changes both the graph representation and the initialization (Li, 2026); Appendix C.1 gives our exclusion rationale. MAP IT provides a further comparison point based on a different representation-matching objective (Jenssen, 2024).

What counts as a better embedding. Embedding fidelity has distinct local, global, density, and inter-cluster aspects, with rankings depending on preprocessing, parameters, datasets, and metrics (Huang et al., 2022). Fidelity, fixed-seed bitwise repeatability, run-to-run geometric stability, and local perturbation stability (Jung et al., 2026) address different questions. Visual artifacts and unreliable individual projections further limit interpretation (Bergam et al., 2026; Xia et al., 2024); even diverse metrics do not establish projection accuracy (Machado et al., 2025).

3 ibUMAP: Coherent and Scalable Field Evaluation

ibUMAP retains UMAP's graph construction and initialization algorithms and modifies only the layout-optimization stage. It keeps UMAP's attraction kernel and parameter conventions (McInnes et al., 2020), replaces stochastic negative-sampling repulsion with a dense field evaluated by interpolation-based FFT (ibFFT), and applies both forces synchronously.

3.1 Preliminaries and the degree-weighted repulsive field

Let $\{x_i\}_{i=1}^n \subset \mathbb{R}^D$ be the input and $y_i \in \mathbb{R}^2$ its embedding, and write $s_{ij} = \|y_i - y_j\|_2^2$. UMAP builds a symmetric fuzzy graph $V = (v_{ij})$ from local neighborhoods, with weighted degree $d_i^{(v)} = \sum_j v_{ij}$ and retained directed entries $\mathcal{E}$; the low-dimensional membership is $w_{ij} = (1 + as_{ij}^b)^{-1}$, with $a, b > 0$ fitted from `min_dist` and `spread` (McInnes et al., 2020). For a retained edge the optimizer applies the attractive force f_{ij}^a, and a negative pair the repulsive kernel K_{ij}:

$$f_{ij}^a = -\frac{2ab\, s_{ij}^{b-1}}{1 + as_{ij}^b}(y_i - y_j), \qquad K_{ij} = \frac{2\gamma b}{(\varepsilon + s_{ij})(1 + as_{ij}^b)}, \tag{1}$$

where γ is the repulsion strength and $\varepsilon > 0$ regularizes short distances (McInnes et al., 2020).

For a fixed embedding Y, uniform negative sampling induces a deterministic conditional expectation of the sampled repulsive update. In the reference implementation this expectation depends on each head's scheduled negative-event count and on coordinate-wise clipping; Appendix A derives it and audits the shared conventions. Motivated by degree-dependent repulsion in negative-sampling analyses (Damrich & Hamprecht, 2021), ibUMAP instead adopts the radial field

$$f_{i,t}^{r,\mathrm{raw}} = \alpha_t E_i R_i^{\mathrm{raw}}, \qquad R_i^{\mathrm{raw}} = \sum_{j \neq i} K_{ij}(y_i - y_j), \qquad E_i = \frac{m d_i^{(v)}}{2n}, \tag{2}$$

where α_t is the learning rate, m the negative-sample rate, and the factor $1/2$ offsets the head-wise attraction of Section 3.3, preserving the reference's expected repulsion-to-attraction ratio. This field replaces the reference implementation's time-dependent event weights and omits its per-interaction coordinate-wise clipping, so it is distinct from the exact expectation of that implementation's scheduled update; it also evaluates all interactions at one snapshot, whereas online in-place updates use successively modified coordinates. The kernel-capped field additionally limits the scalar kernel,

$$K_{ij}^{\mathrm{cap}} = \min\{c_K, K_{ij}\}, \qquad R_i^{\mathrm{cap}} = \sum_{j \neq i} K_{ij}^{\mathrm{cap}}(y_i - y_j), \tag{3}$$

with $c_K = 4$. The event expectation, the raw field, and the kernel-capped field are three different objects, with direct summation as the quadratic reference for the last. Kernel capping changes the field; ibFFT changes only how it is evaluated.

3.2 THREE-MOMENT FIELD EVALUATION

Viewing a dense nonlinear-embedding gradient as an N-body sum follows prior work (Vladymyrov & Carreira-Perpiñán, 2014). For either direct kernel $K^\star \in \{K, K^{\mathrm{cap}}\}$, the degree-weighted vector sum separates as $R_i^\star = y_i S_0(y_i) - S_1(y_i)$ with $S_0(y_i) = \sum_j K_{ij}^\star$ and $S_1(y_i) = \sum_j K_{ij}^\star y_j$. In two dimensions $S_1 = (S_x, S_y)^\top$, so the field is reconstructed from three scalar moments:

$$R_i^\star = \begin{bmatrix} y_{i,x} S_0(y_i) - S_x(y_i) \\ y_{i,y} S_0(y_i) - S_y(y_i) \end{bmatrix}. \tag{4}$$

Every source therefore deposits the three charges $(1, y_{j,x}, y_{j,y})$, which interact through the same translation-invariant scalar kernel. Self-interaction needs no explicit correction: point i enters S_0 and S_1 with the same kernel weight, so its own term cancels exactly in equation 4.

Following the interpolation-based P2M–FFT–M2P pattern established for FIt-SNE (Linderman et al., 2019), ibFFT covers the current embedding by an adaptive square mesh with B boxes per dimension and p interpolation nodes per box and dimension. The mesh is rebuilt each epoch from the coordinate range, and its box width is rounded upward onto a logarithmic scale so that nearby epochs share a quantized width and can reuse a cached kernel spectrum; Appendix C.1 states the box-count rule and default grid settings. Let $\mathcal{G}(j)$ be the p^2 nodes associated with point j and λ_{jg} their tensor-product Lagrange weights. Particle-to-mesh deposition constructs, for $\ell \in \{0, x, y\}$,

$$W_\ell(g) = \sum_{j : g \in \mathcal{G}(j)} \lambda_{jg} \phi_{j,\ell}, \quad \phi_{j,0} = 1, \quad \phi_{j,x} = y_{j,x}, \quad \phi_{j,y} = y_{j,y}.$$

The chosen scalar kernel is sampled at mesh displacements and embedded into a block-circulant array of side length $L = 2pB$ by mirroring and zero-padding, which yields a linear convolution without wraparound in the active region. With $M = L^2$ padded nodes and $\mathcal{K}$ the padded sampled kernel, the three mesh fields are $V_\ell = \mathcal{F}^{-1}[\mathcal{F}(\mathcal{K}) \odot \mathcal{F}(W_\ell)]$. Mesh-to-particle interpolation then gives $\widetilde{S}_\ell(y_i) = \sum_{g \in \mathcal{G}(i)} \lambda_{ig} V_\ell(g)$, after which equation 4 reconstructs $\widetilde{R}_i$.

All reported configurations use $p = 1$. Deposition is then an $O(n)$ accumulation into the box containing each point and mesh-to-particle interpolation is a lookup, so the scheme is piecewise constant: points sharing a box interact at the zero-displacement kernel value $2\gamma b/\varepsilon \approx 1.8 \times 10^3$ (at `min_dist=0.1`), which the cap c_K bounds. The implementation supports general p, but raising it does not improve fidelity in our setting and costs runtime (Appendix B), so $p > 1$ is not used.

3.3 SYNCHRONOUS UPDATES AND SAFEGUARDS

ibUMAP updates coherently: at epoch t, it evaluates attraction and repulsion from the same snapshot $Y^{(t)}$ and applies both only after that evaluation. Unlike in-place updates, no force sees coordinates already moved earlier in the epoch. Scheduled fuzzy-graph edges contribute equation 1 with coordinate-wise gradient clipping; we denote the aggregated attractive contribution, including the learning-rate factor α_t, by $f_{i,t}^a$, and the selected direct or ibFFT field supplies $f_{i,t}^r$. Production ibUMAP aggregates attraction at the head of each scheduled directed edge, whereas the controlled synchronous variants in Section 4 retain the reference two-endpoint attraction update. The aggregation convention and update order are part of the optimizer, in addition to the field operator itself.

Beyond kernel capping in equation 3, the safeguarded configuration norm-clips the repulsive update to $\tau_t = 4\alpha_t$ and damps attraction at high-degree graph hubs by a factor $\eta_i \in (0, 1]$ derived from the 99th percentile of weighted degree (Appendix A). The guarded synchronous update is

$$y_i^{(t+1)} = y_i^{(t)} + \eta_i f_{i,t}^a + \mathrm{clip}_{\|\cdot\|_2 \le \tau_t}\left(f_{i,t}^r\right), \qquad \alpha_{t+1} = \alpha_0 \left(1 - \frac{t}{T}\right). \tag{5}$$

The safeguards matter because online UMAP clips and applies each sampled interaction at once, whereas a synchronous step applies summed forces. Without the norm clip, degree-weighted repulsion ejects hubs at the first, attraction-free update; without damping, summed attraction throws hubs out of the main body (Appendix B). Section 4 measures the associated final-quality change.

3.4 Execution paths, complexity, and repeatability

The CPU path uses reusable FFT plans and buffers, cached kernel spectra, and a fixed-order accumulation mode for repeatable execution. The GPU path batches the three convolution channels, reuses plans and spectra, and fuses reconstruction and update kernels where possible; FFT/GPU force-field evaluation is related to prior graph-layout systems (Zhong et al., 2024), while cuML provides the primary accelerated UMAP baseline (Nolet et al., 2021).

Per epoch, P2M and M2P cost $O(np^2)$, the batched convolutions cost $O(M \log M)$, and scanning the retained fuzzy graph costs $O(|\mathcal{E}|)$, giving optimizer cost $O(T[|\mathcal{E}| + np^2 + M \log M])$ in time and $O(n+|\mathcal{E}|+np^2+M)$ in memory. For $p = 1$, a sparse k-NN graph, and the nominal $M = \Theta(n)$ mesh regime, this is $O(T[nk + n \log n])$, versus $O(Tn^2)$ for the direct all-pairs field. This covers only the optimizer: graph construction and initialization are separate pipeline costs. Sampled UMAP has a different, event-count complexity, so this bound does not establish an asymptotic advantage over negative sampling; relative end-to-end performance is assessed empirically in Section 5.

Because the field formulation removes random negative draws, scheduled attraction, fixed-order P2M accumulation, and repeatable accelerator reductions give repeatable execution without stochastic pair selection. We define same-device repeatability under a fixed backend, precision, and reduction mode; cross-device numerical agreement depends additionally on hardware, FFT libraries, compilers, and numeric types. Section 5 evaluates end-to-end performance, the cost of the seeded execution profile, embedding fidelity, and run-to-run stability.

4 How Each Mechanism Changes Embedding Quality

We trace how the changes from online sampled UMAP to field-based optimization affect final embedding quality, comparing eight variants on 59 datasets in the fixed-input collection, spanning tabular, image, and single-cell data with 150–94,415 observations. All variants reuse the same fuzzy graph and spectral initialization within each dataset and run for 200 epochs with 15 neighbors and `min_dist=0.1`. We average scores over three optimizer seeds within each dataset, then weight datasets equally. Appendix B gives the protocol and exclusion reason, variant means (Table 3), paired-effect summaries, and a check that weights the data families equally. We measure local fidelity by trustworthiness (TW), continuity (C), and neighborhood preservation (NP) at $k = 15$ (Venna & Kaski, 2006; Lee & Verleysen, 2009; Huang et al., 2022), and global fidelity by random triplet accuracy (RTA) and input-to-embedding distance Spearman correlation (ρ_D) (Huang et al., 2022); Appendix C.3 defines them and the shared sampling protocol.

The controlled sequence makes updates synchronous (A→B), then replaces random negative draws with their frozen-state expectation under uniform sampling, retaining scheduled event counts and component-wise clipping (B→C). D instead uses the degree-weighted radial field in equation 2, changing both the weighting and the clipping convention. E–G successively add the scalar kernel cap, default FFT evaluation, and repulsion-norm clipping with attraction damping. B–G keep two-endpoint attraction, under which E_i halves the reference repulsion-to-attraction ratio; H restores it with head-wise attraction and a different execution path, so G→H is not single-factor (Appendix A).

Synchrony and repulsion formulation. Synchrony lowers TW on all 59 datasets and NP on 57 (A→B; Figure 2). The event expectation recovers about 47% of the mean NP loss (B→C). The raw degree-weighted field lowers both global scores on most datasets (C→D); the kernel cap reverses this on 54 and 53 datasets and raises mean C from 0.9470 to 0.9813, but lowers NP on 54 (D→E), trading neighbor overlap for other local and global fidelity measures.

FFT evaluation and update safeguards. FFT evaluation leaves all five paired distributions centered at zero, changing each mean by at most 0.0005 in the local scores and 0.0016 in ρ_D (E→F). Its near-field error arises from pairs sharing a mesh box, which interact at the zero-displacement kernel

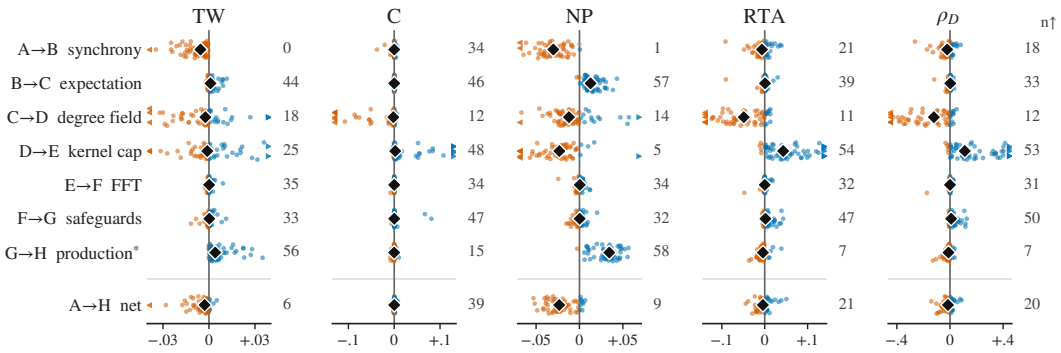


Figure 2: Paired per-dataset quality changes (later minus earlier variant; 59 datasets, seed-averaged; higher is better). Blue improves, orange worsens; diamonds are medians; numbers at right count improving datasets; triangles mark points beyond each 97.5th-percentile axis range. *G→H changes attraction update and execution path; A→H is the net change from UMAP. Means: Table 3.

value and are therefore bounded by the cap c_K introduced in E (Section 3.2). Repulsion-norm clipping and attraction damping improve C, RTA, and ρ_D on most datasets, raising mean ρ_D by 0.0228 at a mean NP change of -0.0002 (F→G); Appendix B shows the failures they prevent.

Relation to the complete optimizer. Production ibUMAP improves NP over G on 58/59 datasets but lowers both global scores on 52/59 (G→H). Relative to UMAP, it matches C but lowers mean TW, NP, RTA, and ρ_D by 0.0051, 0.0209, 0.0037, and 0.0179 (A→H): close to UMAP, but not equivalent. Section 5 shows what this cost buys: faster fits and repeatable parallel CPU execution.

5 End-to-End Performance, Fidelity, and Repeatability

We assess the practical trade-off between end-to-end speed, embedding fidelity, and repeatability when ibUMAP is used in a complete pipeline.

5.1 Benchmark setup

We benchmark 71 datasets with 150 to 4.17 million observations, spanning tabular data, images, word vectors, and single-cell measurements. We compare ibUMAP CPU with umap-learn (McInnes et al., 2018), and ibUMAP CUDA with cuML UMAP (Nolet et al., 2021) and TorchDR UMAP (Van Assel et al., 2026). Runtime and stability comparisons retain all available pairs: 66/71/66 datasets, respectively, with five large datasets resource-skipped for CPU methods and TorchDR. The main fidelity table uses the common 66 datasets; the appendix reports all pairs.

All methods produce two-dimensional embeddings with 15 neighbors, `min_dist=0.1`, and a requested 200-epoch budget under two profiles, *unseeded* and *seeded* (seed 42; deterministic ibUMAP execution). We time five runs per dataset, method, and profile after untimed warmup, including graph construction, initialization, optimization, and synchronized host materialization. Speed ratios use five-run median times and cover both profiles; main fidelity and stability comparisons are unseeded. Fidelity metrics share one sample of at most 10^6 points (seed 42) across methods and runs. Appendix C.1 details timing, execution policies, hardware, and exclusions.

5.2 End-to-end performance

CPU performance. Median speedups over umap-learn are $3.29\times$ unseeded and $5.79\times$ seeded (Figure 3a). Both profiles win on the same 60/66 datasets and lose only below 1,000 observations.

GPU performance. Unseeded ibUMAP trails cuML at the suite median (speed ratio 0.875) but wins on all 11 datasets above 10^5 observations, reaching a median $1.44\times$ speedup across six million-scale datasets. Against TorchDR, it wins on 62/66 datasets (median $1.53\times$ speedup; Figure 3b).

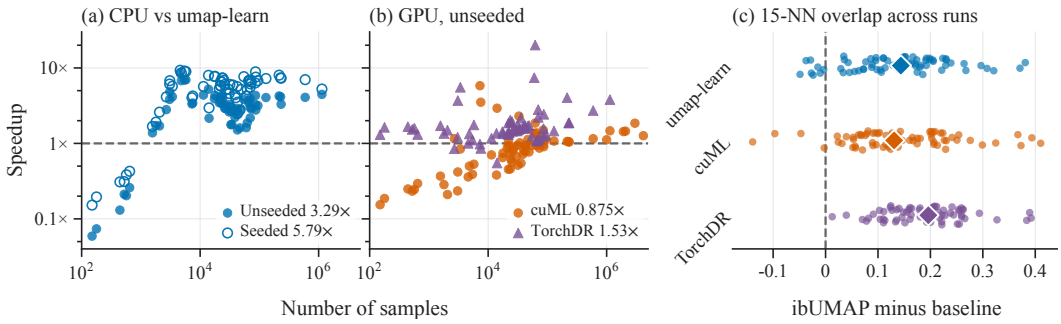


Figure 3: End-to-end speed and run-to-run stability. **(a)** CPU speedup over umap-learn under matching unseeded and seeded profiles and **(b)** unseeded GPU speedup over cuML and TorchDR: points are baseline/ibUMAP ratios of five-run median times and the legends give cross-dataset medians. **(c)** Paired 15-NN overlap differences across unseeded reruns on the same 66/71/66 pairs: positive values favor ibUMAP and diamonds mark cross-dataset medians. Colors denote the same baseline in all panels; only vertical positions in (c) are jittered.

Cost of seeded execution. Both CPU implementations yield bitwise-identical embeddings across five seeded runs on all 66 datasets. Median seeded/unseeded time ratios are 0.961 for ibUMAP and 1.664 for umap-learn, including profile-specific threading policies. On the six million-scale GPU datasets, seeded execution substantially increases both ibUMAP and cuML runtimes, leaving ibUMAP only a 1.05× median speedup. Shared cuML graph construction dominates this cost, limiting the effect of optimizer differences (Appendix C.2, Figure 9).

5.3 Embedding fidelity

We use the same five fidelity measures of Section 4, with 10^6 sampled triplets and at most 10^6 point pairs. Table 1 compares all methods on the common datasets under unseeded execution.

Table 1: Unseeded local and global fidelity on the common 66 datasets: medians of dataset-level five-run mean scores. Metrics match Section 4; higher is better. Bold marks the highest displayed score per CPU/GPU block, including rounding ties, not statistical significance.

Platform	Method	TW ↑	C ↑	NP ↑	RTA ↑	ρ_D ↑
CPU	umap-learn	**0.9901**	0.9942	**0.3699**	**0.7249**	**0.6072**
	ibUMAP	0.9853	**0.9950**	0.3436	0.7023	0.5493
GPU	cuML	**0.9892**	**0.9950**	0.3635	0.7349	0.6295
	TorchDR	0.9873	0.9943	**0.3768**	**0.7647**	**0.6706**
	ibUMAP	0.9851	**0.9950**	0.3407	0.7217	0.5897

ibUMAP's TW, NP, RTA, and ρ_D medians are lower, while C remains close, as in Section 4. These marginal medians do not establish per-dataset advantages; paired differences, seeded scores, full coverage, and collection-weighting checks are in Appendix C.3. In return, ibUMAP fits faster on CPU, reruns more stably (Section 5.4), and keeps its speed when seeded, unlike umap-learn. The BRAQUE-derived case (Section 6) illustrates the lower CPU cost of obtaining fixed-seed repeatable embeddings and clustering. However, higher unseeded geometric stability does not translate into more consistent clustering in this case. Whether the runtime benefit justifies the fidelity trade-off depends on downstream requirements.

5.4 Run-to-run embedding stability

Unseeded outputs are not bitwise identical. We measure their geometric stability through run-to-run 15-nearest-neighbor overlap and sampled pairwise-distance Spearman correlation, using at most 10,000 points per comparison (Appendix C.4).

ibUMAP improves neighbor overlap on 58/66 datasets versus umap-learn, 67/71 versus cuML, and 66/66 versus TorchDR (Figure 3c). Distance correlation improves on most datasets. These gains concern geometric agreement; they imply neither input-space fidelity nor repeatable downstream partitions. Section 6 traces how residual variation propagates to clustering.

6 Reusing Repeatable Embeddings

An embedding can be an input to further computation as well as a visualization. In this setting, geometric similarity across reruns may be insufficient: downstream clustering can change even when the overall layout remains similar. We examine this effect in a UMAP–HDBSCAN workflow and measure the CPU cost of making its outputs repeatable.

6.1 Downstream clustering in a BRAQUE-derived workflow

BRAQUE uses Lognormal Shrinkage preprocessing, UMAP, and HDBSCAN clustering to aid expert interpretation of single-cell protein measurements (Dall'Olio et al., 2023; McInnes et al., 2017). We use its public L2 sample with 56,962 cells and 62 selected features. Feature selection uses the public antibody table as a proxy for the paper-specific reference; this is a BRAQUE-derived computational case, not an exact reproduction of its biological analysis. We freeze the preprocessed input and rerun the full embedding fit in two dimensions with 50 neighbors, `min_dist=0`, and 200 epochs. HDBSCAN settings stay fixed. Each of four CPU configurations—UMAP and ibUMAP, with and without a fixed seed—has five runs. Appendix D gives the protocol and full summary.



(a) Run 1 (b) Run 2 (c) Assignment changes

Figure 4: The first two scheduled unseeded UMAP runs on the same frozen input, colored by HDBSCAN assignments. Cluster colors are matched by maximum cell overlap; gray denotes noise in (a,b). Run 2 is rigidly aligned to Run 1 for display, without scaling; clustering uses the original coordinates. Panel (c) locates assignment changes on Run 1, distinguishing changes between clusters from transitions to or from noise. All 56,962 cells are shown on a common coordinate scale. This pair has distance correlation $\rho = 0.935$, ARI 0.615, and 19.78% changed assignments.

Across the ten unordered pairs of unseeded UMAP runs, sampled pairwise-distance Spearman correlation has median 0.937, yet the median adjusted Rand index (ARI) is 0.611 and the median changed-assignment fraction is 20.80% after maximum-overlap, one-to-one label matching. Noise is kept separate from ordinary clusters. Figure 4 shows the first two scheduled runs, selected without inspecting their discrepancy: 19.78% of assignments change, including 3.49% of cells switching noise status. Cluster counts across the five runs range from 37 to 45. These changes include splits and merges; they measure computational partition variation, not biological misclassification. Even ibUMAP's higher unseeded distance correlation (0.999) coexists with 22.01% changed assignments, underscoring that geometric agreement alone does not ensure repeatable clustering.

6.2 Cost of repeatable reuse

With seed 42 fixed, both methods produce bitwise-identical coordinates and identical HDBSCAN labels across their five runs, but at very different cost. Parallel umap-learn updates shared coordinates in a scheduling-dependent order, so seeding it falls back to one job and serial optimization to fix that order; its median embedding fit rises from 15.44 s to 49.06 s, a 3.18× slowdown. ibUMAP applies

each epoch's updates from a single snapshot and accumulates the field in a fixed order (Section 3.3), so parallelism does not affect the result: its deterministic mode keeps eight jobs and takes 7.10 s, matching its unseeded time (ratio 1.001). Seeded ibUMAP is therefore $6.91\times$ faster than seeded UMAP, or $6.42\times$ including the HDBSCAN fit (7.74 s versus 49.70 s); times exclude preprocessing and warmup. The speedup comes with a fidelity trade-off (Appendix D).

Repeatability also makes visual comparisons controlled. A clustering threshold can be adjusted on a cached embedding; an embedding parameter requires a refit, and with a fixed seed and deterministic execution each setting returns the same layout and partition on every request. Differences between settings then reflect the parameter rather than rerun variation, which here already changes about a fifth of the assignments. ibUMAP lowers the cost of each such refit from 49.06 s to 7.10 s.

7 Limitations and Discussion

Scope of the method. The present ibFFT construction is specialized to two-dimensional output, whereas UMAP itself is not restricted to visualization dimensions (McInnes et al., 2020). Extending the field to higher dimension increases mesh and convolution costs and requires a separate accuracy and scaling study. With fixed FFT and grid-management overhead, sampled UMAP can remain preferable below the empirical crossover or when a specific fidelity metric dominates throughput.

Scope of the repeatability claim. Same-device repeatability is narrower than portable reproducibility: bitwise agreement can depend on the backend, FFT library, compiler, numeric type, and reduction order, and it does not establish GhostUMAP2-style local stability under small initialization and negative-sampling perturbations (Jung et al., 2026). Performance and fidelity conclusions are likewise specific to the tested hardware and dataset regimes.

Scope of the evidence. Synchronous aggregation changes optimizer semantics as well as computation. Section 4 measures its final-quality effects under one fixed graph and spectral initialization, leaving robustness to other initializations untested, and whether the dynamical predictions of attraction-shape theory for pairwise UMAP (Islam & Fleischer, 2026) transfer to synchronous updates remains open. Its 59 datasets are also not independent: several single-cell subsets share a source resource, dataset size and source family are associated, and the family-weighting check limits but does not remove this dependence. We characterize the ibFFT approximation by its effect on final embeddings and by an interpolation-order sweep, rather than by field-level error against the direct sum. The downstream study uses a single BRAQUE-derived proxy workflow without expert phenotype labels; it demonstrates computational partition repeatability and its cost, while generalization across domains and effects on human interpretation remain untested.

Interpreting the output, and what comes next. One interpretive limit bears directly on ibUMAP's own output: because kernel capping and synchrony change how tightly clusters contract, area and compactness in these layouts should not be read as input density without a corresponding diagnostic (Narayan et al., 2021). The broader opportunity is to ask when other sampled interaction kernels admit a useful expected-field realization. Each such extension requires its own estimator audit, field decomposition, numerical reference, and optimizer study.

8 Conclusion

ibUMAP makes UMAP optimization coherent: it evaluates attraction and a degree-weighted repulsive field from one snapshot and applies them synchronously. Three scalar moments and interpolation-based FFT make this dense field cheap to evaluate on CPUs and GPUs. In fixed-graph, fixed-initialization comparisons on 59 datasets, this synchrony costs local fidelity, kernel capping trades neighbor overlap for continuity and global fidelity, and FFT evaluation changes average quality little. Coherence allows seeded CPU runs to stay parallel yet bitwise repeatable on the same device, at no median runtime penalty. End-to-end benchmarks yield median CPU speedups of $3.29\times$ unseeded and $5.79\times$ seeded, and a $1.44\times$ unseeded GPU speedup over cuML on million-scale datasets. In unseeded runs, ibUMAP has lower TW, NP, and global-fidelity medians but greater run-to-run stability. The BRAQUE-derived case shows that geometric stability does not ensure repeatable clustering, while ibUMAP reduces the CPU cost of repeatable fits.

AI USE STATEMENT

We used generative AI tools (conversational assistants, coding assistants, and agentic coding tools) for the following tasks with required disclosure. *Theoretical formulation.* We derived the expected repulsive field, its three-moment evaluation, and the UMAP kernel for interpolation-based FFT from prior work (McInnes et al., 2020; Linderman et al., 2019; Zhong et al., 2024); AI tools unified the notation and wording across our drafts of these formulations (Section 3). *Derivations.* AI tools derived the expected event counts and repulsion-to-attraction ratios in Appendix A and implemented the accompanying audit of the reference sampling schedule; we checked the derivations and the audit results. *Hypotheses and experimental design.* We hypothesized that quality differences between ibUMAP and UMAP arise from synchronous updates, FFT approximation, and safeguards, and designed the asynchronous control, the fixed-graph and fixed-initialization protocol, and the end-to-end benchmark. AI tools helped refine this hypothesis into separate factors, including the change of field formulation, and proposed variants C and D of the ablation sequence (Section 4). *Implementation and data preparation.* We wrote the UMAP kernel and the overall architecture ourselves, adapting the FFT kernels of Zhong et al. (2024) and the umap-learn pipeline. AI tools migrated and consolidated this code into the released repository and unified our earlier dataset download and preprocessing scripts. The live demo of the downstream case was implemented by AI (Appendix D). We did not use generative AI tools to generate synthetic datasets, to formulate the main claims (replacing negative sampling with an expected field and evaluating it by FFT), or to select evaluation metrics. Translation and qualitative data analysis are not applicable to this work. Additionally, we used AI tools to search for and summarize related work, including identifying the BRAQUE case, to brainstorm and organize ideas, to edit code, to create and revise figures, and to draft and polish parts of the paper. In the supplementary video, the narration was synthesized by an AI text-to-speech tool from a script that AI tools helped edit; the slides, animations, and demo screen recordings were made by us without generative AI. The interpretations of results are ours; AI tools drafted part of the interpretive text, which we reviewed and revised. The authors reviewed all AI-assisted work and checked every cited reference against the original publication. All reported numbers, including runtimes and quality scores, are produced by experiment scripts from recorded run outputs. We verified the migrated code with the repository's unit and protocol tests, by checking that it rebuilds the fixed inputs of the mechanism study bitwise and rescores a sample of stored embeddings to the reported values, and by regenerating all result tables and figures from the frozen run summaries. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work involves no human subjects and no new data collection. All datasets are previously published and publicly available; human-derived single-cell data are used only in the form released by their providers. Our repository provides scripts that download the data from their original sources rather than redistributing them. Faster or bitwise-repeatable embeddings are not thereby more faithful: ibUMAP retains fidelity trade-offs (Section 5.3), and two-dimensional embeddings can mislead interpretation (Section 7).

REPRODUCIBILITY STATEMENT

Section 3 specifies ibUMAP, and Section 3.4 its execution paths and repeatability modes. Appendix A derives the expected field and audits it against the reference implementation. Appendices B, C, and D give the protocols of the mechanism study, the frozen end-to-end benchmark (including software versions, hardware, and the complete dataset manifest), and the downstream case. The anonymous repository `https://anonymous.4open.science/r/ibUMAP-522A` contains the ibUMAP source code, environment specifications, data download and preprocessing scripts, experiment and plotting scripts, and the result tables from which the reported numbers and figures are generated, as well as a live demo of the downstream case (Appendix D).

REFERENCES

Stefan Aeberhard and Michele Forina. Wine. UCI Machine Learning Repository, 1991. URL https://doi.org/10.24432/C5PC7J.

Ethem Alpaydin and Cenk Kaynak. Optical recognition of handwritten digits. UCI Machine Learning Repository, 1998. URL https://doi.org/10.24432/C50P49.

Ralph G. Andrzejak, Klaus Lehnertz, Florian Mormann, Christoph Rieke, Peter David, and Christian E. Elger. Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state. *Physical Review E*, 64(6):061907, 2001. doi: 10.1103/PhysRevE.64.061907.

Martin Aumüller, Erik Bernhardsson, and Alexander Faithfull. ANN-Benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. *Information Systems*, 87:101374, 2020. doi: 10.1016/j.is.2019.02.006.

Noah Bergam, Szymon Snoeck, and Nakul Verma. t-SNE exaggerates clusters, provably. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=rrbKiGMA5s.

Jan Niklas Böhm, Philipp Berens, and Dmitry Kobak. Attraction-repulsion spectrum in neighbor embeddings. *Journal of Machine Learning Research*, 23(95):1–32, 2022. URL https://www.jmlr.org/papers/v23/21-0055.html.

Xin Chen, Shuowei Hou, Yifan Wang, Mingliang Xue, Zezheng Feng, Oliver Deussen, Weidong Huang, and Yunhai Wang. SNAP-tFDP: Massively scalable graph layouts via sparse negative sampling. arXiv preprint arXiv:2608.01907, 2026. URL https://arxiv.org/abs/2608.01907.

Eduardo da Veiga Beltrame. Packer et al. 2019 scRNA-seq dataset wrangled into standard WormBase AnnData. CaltechDATA, 2021. URL https://doi.org/10.22002/D1.1945.

Lorenzo Dall'Olio, Maddalena Bolognesi, Simone Borghesi, Giorgio Cattoretti, and Gastone Castellani. BRAQUE: Bayesian reduction for amplified quantization in UMAP embedding. *Entropy*, 25(2):354, 2023. doi: 10.3390/e25020354.

Sebastian Damrich and Fred A. Hamprecht. On UMAP's true loss function. In *Advances in Neural Information Processing Systems*, volume 34, pp. 5798–5809, 2021.

Sebastian Damrich, Jan Niklas Böhm, Fred A. Hamprecht, and Dmitry Kobak. From t-SNE to UMAP with contrastive learning. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=B8a1FcY0vi.

Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *The Annals of Statistics*, 32(2):407–499, 2004. doi: 10.1214/009053604000000067.

Mateus Espadoto, Rafael M. Martins, Andreas Kerren, Nina S. T. Hirata, and Alexandru C. Telea. Toward a quantitative survey of dimension reduction techniques. *IEEE Transactions on Visualization and Computer Graphics*, 27(3):2153–2173, 2021. doi: 10.1109/TVCG.2019.2944182.

Ronald A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7 (2):179–188, 1936. doi: 10.1111/j.1469-1809.1936.tb02137.x.

Isabelle Guyon, Amir Saffari, Gideon Dror, and Gavin Cawley. Agnostic learning vs. prior knowledge challenge. In *Proceedings of the International Joint Conference on Neural Networks*, pp. 829–834, 2007. doi: 10.1109/IJCNN.2007.4371065.

Mark Hopkins, Erik Reeber, George Forman, and Jaap Suermondt. Spambase. UCI Machine Learning Repository, 1999. URL https://doi.org/10.24432/C53G6X.

Haiyang Huang, Yingfan Wang, Cynthia Rudin, and Edward P. Browne. Towards a comprehensive evaluation of dimension reduction methods for transcriptomic data visualization. *Communications Biology*, 5:719, 2022. doi: 10.1038/s42003-022-03628-x.

Mohammad Tariqul Islam and Jason W. Fleischer. The shape of attraction in UMAP: Exploring the embedding forces in dimensionality reduction. *Transactions on Machine Learning Research*, 2026. URL https://openreview.net/forum?id=fdPNhqav5G.

Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2011. doi: 10.1109/TPAMI.2010.57.

Robert Jenssen. MAP IT to visualize representations. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=OKf6JtXtoy.

Brian Johnson. Forest type mapping. UCI Machine Learning Repository, 2012. URL https://doi.org/10.24432/C5QP56.

Myeongwon Jung, Takanori Fujiwara, and Jaemin Jo. GhostUMAP2: Measuring and analyzing (r, d)-stability of UMAP. *IEEE Transactions on Visualization and Computer Graphics*, 32(1): 353–362, 2026. doi: 10.1109/TVCG.2025.3633894.

Dmitry Kobak and George C. Linderman. Initialization is critical for preserving global data structure in both t-SNE and UMAP. *Nature Biotechnology*, 39:156–157, 2021. doi: 10.1038/s41587-020-00809-z.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

John A. Lee and Michel Verleysen. Quality assessment of dimensionality reduction: Rank-based criteria. *Neurocomputing*, 72(7–9):1431–1443, 2009. doi: 10.1016/j.neucom.2008.12.017.

Hongmin Li. FastUMAP: Scalable dimensionality reduction via bipartite landmark sampling. arXiv preprint arXiv:2605.11428, 2026. URL https://arxiv.org/abs/2605.11428.

George C. Linderman, Manas Rachh, Jeremy G. Hoskins, Stefan Steinerberger, and Yuval Kluger. Fast interpolation-based t-SNE for improved visualization of single-cell RNA-seq data. *Nature Methods*, 16(3):243–245, 2019. doi: 10.1038/s41592-018-0308-4.

Alister Machado, Michael Behrisch, and Alexandru Telea. Necessary but not sufficient: Limitations of projection quality metrics. *Computer Graphics Forum*, 44(3):e70101, 2025. doi: 10.1111/cgf.70101.

Michael McCann and Adrian Johnston. SECOM. UCI Machine Learning Repository, 2008. URL https://doi.org/10.24432/C54305.

Leland McInnes, John Healy, and Steve Astels. hdbscan: Hierarchical density based clustering. *Journal of Open Source Software*, 2(11):205, 2017. doi: 10.21105/joss.00205.

Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. UMAP: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29):861, 2018. doi: 10.21105/joss.00861.

Leland McInnes, John Healy, and James Melville. UMAP: Uniform manifold approximation and projection for dimension reduction. arXiv preprint arXiv:1802.03426, 2020. URL https://arxiv.org/abs/1802.03426.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, volume 26, pp. 3111–3119, 2013.

Sérgio Moro, Paulo Cortez, and Paulo Rita. A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62:22–31, 2014. doi: 10.1016/j.dss.2014.03.001.

Ashwin Narayan, Bonnie Berger, and Hyunghoon Cho. Assessing single-cell transcriptomic variability through density-preserving data visualization. *Nature Biotechnology*, 39:765–774, 2021. doi: 10.1038/s41587-020-00801-7.

Corey J. Nolet, Victor Lafargue, Edward Raff, Thejaswi Nanditale, Tim Oates, John Zedlewski, and Joshua Patterson. Bringing UMAP closer to the speed of light with GPU acceleration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 418–426, 2021. doi: 10.1609/aaai.v35i1.16118.

Jonathan S. Packer, Qin Zhu, Chau Huynh, Priya Sivaramakrishnan, Elicia Preston, Hannah Dueck, Derek Stefanik, Kai Tan, Cole Trapnell, Junhyong Kim, Robert H. Waterston, and John I. Murray. A lineage-resolved molecular atlas of *C. elegans* embryogenesis at single-cell resolution. *Science*, 365(6459):eaax1971, 2019. doi: 10.1126/science.aax1971.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. URL https://www.jmlr.org/papers/v12/pedregosa11a.html.

Nicola Pezzotti, Julian Thijssen, Alexander Mordvintsev, Thomas Höllt, Baldur Van Lew, Boudewijn P. F. Lelieveldt, Elmar Eisemann, and Anna Vilanova. GPGPU linear complexity t-SNE optimization. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):1172–1181, 2020. doi: 10.1109/TVCG.2019.2934307.

John P. Ryan, Sebastian E. Ament, Carla P. Gomes, and Anil Damle. The fast kernel transform. In *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pp. 11669–11690. PMLR, 2022. URL https://proceedings.mlr.press/v151/ryan22a.html.

Marek Sikora and Łukasz Wróbel. Seismic-bumps. UCI Machine Learning Repository, 2010. URL https://doi.org/10.24432/C5W902.

W. Nick Street, William H. Wolberg, and Olvi L. Mangasarian. Nuclear feature extraction for breast tumor diagnosis. In *Biomedical Image Processing and Biomedical Visualization*, volume 1905 of *Proceedings of SPIE*, pp. 861–870, 1993. doi: 10.1117/12.148698.

The Tabula Sapiens Consortium. Tabula Sapiens 2.0: A comprehensive transcriptomic atlas of human cell types. *Cell*, 189, 2026. doi: 10.1016/j.cell.2026.08.010. URL https://doi.org/10.1016/j.cell.2026.08.010. In press.

Hugues Van Assel, Nicolas Courty, Rémi Flamary, Aurélien Garivier, Mathurin Massias, Titouan Vayer, and Cédric Vincent-Cuaz. TorchDR: PyTorch dimensionality reduction. Software, version 0.4, 2026. URL https://github.com/TorchDR/TorchDR.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008. URL https://www.jmlr.org/papers/v9/vandermaaten08a.html.

Jarkko Venna and Samuel Kaski. Local multidimensional scaling. *Neural Networks*, 19(6–7):889–899, 2006. doi: 10.1016/j.neunet.2006.05.014.

Max Vladymyrov and Miguel A. Carreira-Perpiñán. Linear-time training of nonlinear low-dimensional embeddings. In *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics*, volume 33 of *Proceedings of Machine Learning Research*, pp. 968–977. PMLR, 2014. URL https://proceedings.mlr.press/v33/vladymyrov14.html.

Qiuyi Wu and Ernest Fokoue. Epileptic seizure recognition. UCI Machine Learning Repository, 2017. URL https://archive.ics.uci.edu/dataset/388/epileptic+seizure+recognition.

Lucy Xia, Christy Lee, and Jingyi Jessica Li. Statistical method scDEED for detecting dubious 2D single-cell embeddings and optimizing t-SNE and UMAP hyperparameters. *Nature Communications*, 15(1):1753, 2024. doi: 10.1038/s41467-024-45891-y.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747, 2017. URL https://arxiv.org/abs/1708.07747.

Meng Zhang, Xingjie Pan, Won Jung, Aaron R. Halpern, Stephen W. Eichhorn, Zhiyun Lei, Limor Cohen, Kimberly A. Smith, Bosiljka Tasic, Zizhen Yao, Hongkui Zeng, and Xiaowei Zhuang. Molecularly defined and spatially resolved cell atlas of the whole mouse brain. *Nature*, 624 (7991):343–354, 2023. doi: 10.1038/s41586-023-06808-9.

Fahai Zhong, Mingliang Xue, Jian Zhang, Fan Zhang, Rui Ban, Oliver Deussen, and Yunhai Wang. Force-directed graph layouts revisited: A new force based on the t-distribution. *IEEE Transactions on Visualization and Computer Graphics*, 30(7):3650–3663, 2024. doi: 10.1109/TVCG.2023.3238821.

A EXPECTED FIELD: DERIVATION AND IMPLEMENTATION AUDIT

This appendix derives the frozen-state expectation of UMAP's sampled repulsive update, relates the degree-weighted field of equation 2 to it, and audits the conventions that the variants of Section 4 share or change. Statements about the reference refer to `umap-learn` 0.5.12 (its sampling calendar and single-epoch Euclidean optimizer); statements about ibUMAP refer to its CPU and CUDA optimizers and to the code of the controlled variants. The calendar statistics in Table 2 replay the reference sampling schedule, without random draws, on the saved optimizer graphs of the 59 mechanism datasets; they involve neither optimization nor quality evaluation.

Graph and edge counting. The optimizer receives the symmetric fuzzy graph after UMAP's pruning step, which removes entries with $v_{ij} < v_{\max}/T$. Each undirected edge is stored as two directed entries $(i, j), (j, i) \in \mathcal{E}$ of equal weight, and ibUMAP and the controlled variants both take $d_i^{(v)}$ as the row sum of the pruned graph. All 59 saved graphs are exactly symmetric, have $v_{\max} = 1$ and $v_{\min}/v_{\max} \geq 1/T = 0.005$, and contain no isolated vertex. An entry has the sampling interval $\tau_{ij} = v_{\max}/v_{ij} \geq 1$: it is active at epoch t once its next-sample time is at most t, after which that time advances by τ_{ij}. Its long-run activation rate is therefore $v_{ij}/v_{\max}$ per epoch, and head i has on average $d_i^{(v)}/v_{\max}$ active entries per epoch.

Activation schedule. Next-sample times start at $\tau_{ij} \geq 1$, so no entry is active at epoch 0; the first active epoch is $t = 1$ on all 59 datasets. Each entry also carries a negative-sample clock σ_{ij} with interval τ_{ij}/m, initialized to τ_{ij}/m. When entry (i, j) is active at epoch t, it draws $\lfloor m(t-\sigma_{ij})/\tau_{ij} \rfloor$ negative samples, and σ_{ij} advances by that many intervals. A first activation thus draws between $m - 1$ and $2m - 2$ negatives, and later activations draw m on average. Summing over the entries with head i gives the scheduled count $c_i(t)$. Its time average $\bar{c}_i$ is close to $md_i^{(v)}/v_{\max}$, slightly lower because epoch 0 is inactive and first activations draw fewer negatives (Table 2). The total count also varies over time: at $t = 1$ only entries with $v_{ij} = v_{\max}$ are active and the total is about 0.17 of its mean over $t \geq 1$, whereas from $t = 10$ onward it stays within 0.91–1.10 of that mean on every dataset.

Negative samples and the event expectation. A negative sample for head i is drawn as $k = \texttt{tau_rand_int} \bmod n$ from a per-head Tausworthe state. We treat k as uniform on all n points, including i, and ignore the modulo bias of the integer generator. A self-draw is skipped but still counts toward $c_i(t)$; a draw at a coincident non-self point contributes zero, because its coefficient is set to zero. Only the head moves under repulsion. The online reference applies each draw to the head's current, already modified coordinates, whereas in the synchronous variant B all draws of an epoch read the snapshot $Y^{(t)}$. Conditional on $Y^{(t)}$ and $c_i(t)$, the expected repulsive update of head i in B is

$$f_{i,t}^{r,C} = \frac{\alpha_t c_i(t)}{n} \sum_{j \neq i} \mathrm{clip}_{[-4,4]}\big(K_{ij}(y_i - y_j)\big), \tag{6}$$

where clipping acts on each coordinate of each interaction and K_{ij} is the kernel of equation 1 with $\gamma = 1$ and the reference's hard-coded $\varepsilon = 10^{-3}$. Variant C evaluates equation 6 exactly, with `float64` accumulation, and reads $c_i(t)$ from the shared sampling calendar before the calendar advances. A unit test enumerates every possible negative target of a single sampling event and reproduces equation 6 to within 10^{-12}.

Update symmetry and attraction. In the reference, an active entry (i, j) moves both endpoints: y_i by $\alpha_t \,\mathrm{clip}_{[-4,4]}(f_{ij}^a)$ and y_j by the negative of that amount. Because (i, j) and (j, i) are both stored and follow the same schedule, each point receives on average $2d_i^{(v)}/v_{\max}$ attractive events per epoch, half as head and half as tail; the calendar replay confirms that head and tail event counts coincide for every point. Variants B–G reuse this two-endpoint update. Production ibUMAP (H) accumulates each active entry only at its head, and the mirrored entry supplies the other endpoint, so each point receives $d_i^{(v)}/v_{\max}$ attractive events per epoch, half the reference count. The attractive coefficient, its coordinate-wise clip at 4, and the parameters a and b, fitted by the reference routine from `min_dist` and `spread`, are identical in all variants.

Normalization of the degree-weighted field. D–H replace the event weight $\alpha_t c_i(t)/n$ by $\alpha_t E_i$ with $E_i = md_i^{(v)}/(2n)$. This weight is constant over epochs and, since $v_{\max} = 1$, about half of the average event weight: $E_i/(\bar{c}_i/n)$ has median 0.507 (Table 2). In expectation, the ratio of repulsive weight to attractive events per point is $(md_i^{(v)}/n)/(2d_i^{(v)}) = m/(2n)$ for the reference, B, and C; $(md_i^{(v)}/(2n))/(2d_i^{(v)}) = m/(4n)$ for D–G; and $(md_i^{(v)}/(2n))/d_i^{(v)} = m/(2n)$ for H. Combined with head-wise attraction, the factor $1/2$ therefore preserves the reference balance between repulsion and attraction and halves both, which in expectation amounts to halving the step size. Under the two-endpoint attraction of D–G, the same factor halves the relative repulsion. Measured on the replayed calendars, this balance relative to the reference has median 0.507 for D–G and 1.014 for H. Consequently, C→D lowers the average repulsive weight by about half, in addition to removing its time dependence and the component-wise clip; D–G run at about half the reference balance; and G→H restores it while also changing the execution path. These are statements about expected event weights. Clipping, the kernel cap, and damping further change the forces, and no variant isolates the factor experimentally.

Table 2: Replayed sampling calendars on the 59 mechanism datasets ($T = 200$, $m = 5$). Per-point quantities are summarized by their median within each dataset; epoch totals are per dataset. The table gives the median and range of these values across datasets.

Quantity	Median	Range
Mean scheduled count, $\bar{c}_i/(md_i^{(v)})$	0.986	0.984–0.987
First activations drawing $m-1$ negatives (share of entries)	0.766	0.728–0.862
Total count at $t = 1$ / mean over $t \geq 1$	0.173	0.163–0.261
Minimum total count over $t \geq 10$ / mean over $t \geq 1$	0.983	0.907–0.989
Maximum total count over $t \geq 1$ / mean over $t \geq 1$	1.016	1.012–1.099
$E_i/(\bar{c}_i/n)$, equal to the D–G balance / reference	0.507	0.507–0.508
Repulsion-to-attraction balance of H / reference	1.014	1.013–1.017

Kernel parameters and clipping. A and B clip each coordinate of each sampled repulsive interaction to $[-4, 4]$, and C applies the same clip inside its expectation. D applies no repulsive clip. E–H cap the scalar kernel at $c_K = 4$ and do not clip individual interactions. The two conventions are not nested: the component clip bounds each interaction's contribution $K_{ij}|y_{i,d} - y_{j,d}|$, whereas the cap bounds K_{ij} and hence limits the contribution to $4\|y_i - y_j\|_2$, which is stricter for nearby points. G and H additionally norm-clip the aggregated repulsive update at $4\alpha_t$ and damp attraction by $\eta_i = \min\{1, (d_{99}/d_i^{(v)})^{1/2}\}$, where d_{99} is the 99th percentile of weighted degree.

Learning-rate schedule and the first update. All variants update the learning rate after each epoch, $\alpha_{t+1} = \alpha_0(1 - t/T)$, so epochs 0 and 1 both use α_0 and epoch $T-1$ uses $2\alpha_0/T$. Because epoch 0 is inactive in the reference and in A–C, their first update occurs at $t = 1$; in C the recorded scheduled count at $t = 0$ is zero on all datasets. D–H evaluate the dense field at every epoch, including $t = 0$, when no attractive entry is active yet: their first update is purely repulsive and uses the full initial learning rate. Production ibUMAP evaluates its field once per epoch, T times in total. Most recorded activity of the repulsion-norm clip in G falls in this update: it clips a median of 26.6% of points (range 0%–95.1% across the 59 datasets), whereas at the later recorded steps, taken every five epochs, the largest clipped share has median 0% and never exceeds 9.7%.

Precision and implementation checks. Embeddings are `float32` in all variants. CPU sampling calendars use `float64`, as in the reference, whereas CUDA calendars use `float32`, which can shift individual activation epochs. Direct field sums in C–E accumulate in `float64`. The asynchronous implementation control reproduces reference UMAP bitwise in all 177 retained dataset–seed comparisons, and the synchronous routine shared by B–G is checked bitwise against the library implementation. The derivation concerns a frozen state and ideal uniform sampling; it does not imply that the trajectories of different variants agree.

B MECHANISM STUDY: PROTOCOL AND ROBUSTNESS

Coverage and weighting. The fixed-input collection contains 60 candidate datasets with at most 10^5 observations. We retain 59: 11 tabular datasets, 4 image datasets, 30 Tabula Sapiens subsets, 4 scDEED datasets, and 10 C. elegans subsets. The retained collection spans 150–94,415 observations and includes Spambase and HIVA. Dataset identifiers and scores are retained in the machine-readable experiment outputs.

We exclude the C. elegans global-QC annotated dataset ($n = 54{,}649$) from all variants and seeds because its asynchronous implementation control fails the bitwise comparison with reference UMAP at seed 2026. Two repeated checks on the original CPU reproduce each implementation's saved output but confirm the disagreement between them; its cause remains unresolved. This exclusion preceded quality evaluation. The raw outputs and diagnostic evidence are retained.

These datasets are not 59 independent data collections: several single-cell subsets share their source resource or observations, and data size is associated with source family. We therefore avoid interpreting coverage differences as isolated size effects. Table 3 weights datasets equally, while Table 7 first averages datasets within each family and then weights the five families equally. The latter is a sensitivity check, not a claim that the families define a uniquely correct target population.

Matched optimization. All variants consume saved fuzzy graphs and spectral initial coordinates constructed with seed 42. Common parameters are two output dimensions, Euclidean distance, 15 neighbors, `min_dist=0.1`, spread 1, learning rate 1, repulsion strength 1, negative-sample rate 5, and 200 epochs. CPU runs use `float32` embeddings, eight configured Numba/FFT threads, `umap-learn` 0.5.12, and FFTW's estimate planner. Direct field sums accumulate in `float64`. The matched variants use the reference post-epoch learning-rate schedule, so epochs 0 and 1 both use the initial learning rate; epoch 0 applies no attraction (Appendix A). The additional asynchronous implementation control is bitwise identical to reference UMAP in all 177 retained dataset–seed comparisons and is not separately scored.

Variant semantics. C uses the ideal-uniform frozen-state expectation equation 6, with the scheduled negative-event count $c_i(t)$ read from the shared sampling calendar before its advancement; Appendix A derives it and audits the shared conventions. D instead uses $\alpha_t m d_i^{(v)}/(2n)$ times the raw radial-field sum, without component-wise clipping. E applies the scalar kernel cap $c_K = 4$. F uses the default piecewise-constant FFT setting ($p = 1$ and box-scale multiplier 1) described in Appendix C.1. G additionally clips the repulsive update at norm $4\alpha_t$ and damps attraction using the 99th-percentile weighted degree and exponent 0.5. Their joint comparison does not isolate the two update safeguards individually. H uses the production optimizer, including its head-wise attraction aggregation. The additional B + repulsion-norm clip branch retains sampled repulsion and adds only the per-epoch repulsion-norm threshold $4\alpha_t$.

Scores and repeated records. Seeds 42, 137, and 2026 vary optimizer draws only; the graph and initialization stay fixed. C–G do not depend on these draws, and their deterministic trajectories are computed once and reused. Within the retained set, the 1,770 optimization records comprise 1,180 executed runs and 590 deterministic reuse records. All five quality metrics cover the 1,593 evaluation records for A–H and the sampled-repulsion clipping control. These counts do not represent independent repetitions for the deterministic variants.

All rows of each dataset enter quality evaluation. TW, C, and NP use 15 neighbors; RTA and ρ_D use the million-triplet/pair protocol with sampling seed 42 in Appendix C.3. The paired-effect and family-weighting tables report all five metrics; the paired table also includes the additional clipping control. We report descriptive comparisons without noninferiority or significance claims.

Table 3: Local (TW, C, NP) and global (RTA, ρ_D) fidelity under a fixed graph and initialization: means across 59 datasets after averaging optimizer seeds within each dataset. Higher is better. H, the production optimizer used in Sections 5 and 6, differs from G in both attraction aggregation and execution path; its net effect is read against A (UMAP), not as one mechanism after G.

Variant	TW ↑	C ↑	NP ↑	RTA ↑	ρ_D ↑
A. UMAP	0.9718	0.9834	0.4281	0.7070	0.5627
B. Synchronous sampled	0.9630	0.9825	0.3973	0.7006	0.5366
C. Synchronous event expectation	0.9648	0.9829	0.4118	0.7001	0.5349
D. Direct degree-weighted field	0.9568	0.9470	0.4010	0.6463	0.3583
E. D + kernel cap	0.9594	0.9813	0.3756	0.7013	0.5396
F. E with default FFT	0.9598	0.9812	0.3760	0.7012	0.5379
G. F + norm clip and damping	0.9599	0.9841	0.3758	0.7083	0.5608
H. Production ibUMAP	0.9667	0.9838	0.4072	0.7033	0.5448

Table 4: Distribution of paired quality changes across 59 datasets. Each cell gives the median signed quality improvement and, in parentheses, the number of datasets with an improvement, computed before rounding. Positive values favor the later variant. A–H match Table 3, and Figure 2 plots the underlying per-dataset changes. The final row is the additional sampled-repulsion clipping control.

Comparison	TW	C	NP	RTA	ρ_D
B–A	-0.0055 (0)	$+0.0001$ (34)	-0.0302 (1)	-0.0064 (21)	-0.0211 (18)
C–B	$+0.0010$ (44)	$+0.0001$ (46)	$+0.0128$ (57)	$+0.0007$ (39)	$+0.0023$ (33)
D–C	-0.0023 (18)	-0.0018 (12)	-0.0121 (14)	-0.0489 (11)	-0.1211 (12)
E–D	-0.0011 (25)	$+0.0022$ (48)	-0.0232 (5)	$+0.0431$ (54)	$+0.1112$ (53)
F–E	$+0.0001$ (35)	$+0.0000$ (34)	$+0.0004$ (34)	$+0.0000$ (32)	$+0.0001$ (31)
G–F	$+0.0002$ (33)	$+0.0001$ (47)	$+0.0002$ (32)	$+0.0017$ (47)	$+0.0089$ (50)
H–G	$+0.0041$ (56)	-0.0001 (15)	$+0.0344$ (58)	-0.0042 (7)	-0.0113 (7)
H–A	-0.0028 (6)	$+0.0001$ (39)	-0.0233 (9)	-0.0043 (21)	-0.0152 (20)
B + norm clip – B	$+0.0001$ (33)	$+0.0001$ (44)	-0.0008 (25)	$+0.0055$ (46)	$+0.0125$ (45)

Complete optimizer versus UMAP. Because H differs from G in more than one factor, its comparison with A gives the net effect of all changes. Production ibUMAP matches UMAP in C (mean $+0.0004$, 39/59 datasets improving) and trails it in the other four metrics: mean TW, NP, RTA, and ρ_D are lower by 0.0051, 0.0209, 0.0037, and 0.0179, improving on 6/59, 9/59, 21/59, and 20/59 datasets (Table 4). The largest gaps, in NP and ρ_D, amount to about 5% and 3% of UMAP's means. Under matched inputs, the complete optimizer is therefore close to UMAP but not equivalent; Section 5 reports the speed and repeatability obtained in exchange.

Heterogeneity of effects. The median paired differences in Table 4 complement the means in the main table. For example, the kernel cap increases mean trustworthiness but decreases its median paired value; it does not improve this metric uniformly. The default FFT comparison has a median absolute change of 0.0018 in NP. Repulsion-norm clipping and attraction damping improve C on 47/59 datasets, while their NP effects are mixed. Adding norm clipping to sampled repulsion alone improves RTA on 46/59 datasets and ρ_D on 45/59, with median gains of 0.0055 and 0.0125.

Safeguard failure cases. Repulsion-norm clipping and attraction damping were introduced after individual runs failed. Figure 5 and Table 5 revisit these runs with the production optimizer H, switching off one of the two safeguards at a time; the kernel cap stays on. CIFAR-10 and scDEED CART reuse the fixed graphs and initializations of this study, and all other settings match H. Under these inputs the three optimizer seeds give bitwise-identical embeddings for every variant, and the complete variant reproduces the H embeddings used above bitwise. We summarize a layout by the ratio of its bounding-box area to that of the 99% of points nearest the coordinate-wise median, and call a point far when its distance to the median exceeds twice that of the 99% quantile. On the 59 datasets, H gives a median ratio of 1.01 and a maximum of 1.52, UMAP a maximum of 1.93, and neither leaves a far point.

Table 5: Safeguard failure cases on the fixed inputs of the mechanism study (production optimizer H, one safeguard switched off at a time; seed 42, as all three seeds give bitwise-identical embeddings). Ratio: bounding-box area of all points over that of the 99% of points nearest the median. Far points lie beyond twice the 99% radius. The peak is the largest ratio after any of the 200 updates, with the update index in parentheses; the last column counts updates after which the ratio exceeds 10.

Dataset	Variant	Final ratio	Far points	Peak ratio	Updates > 10
CIFAR-10	All safeguards	1.46	0	11.8 (7)	12
	No repulsion-norm clip	1,940	555 (0.9%)	1,940 (200)	200
	No attraction damping	1.70	0	90.2 (2)	157
scDEED CART	All safeguards	1.02	0	18.6 (21)	38
	No repulsion-norm clip	93.9	483 (0.8%)	93.9 (200)	146
	No attraction damping	2.08	1 (<0.1%)	296 (17)	175

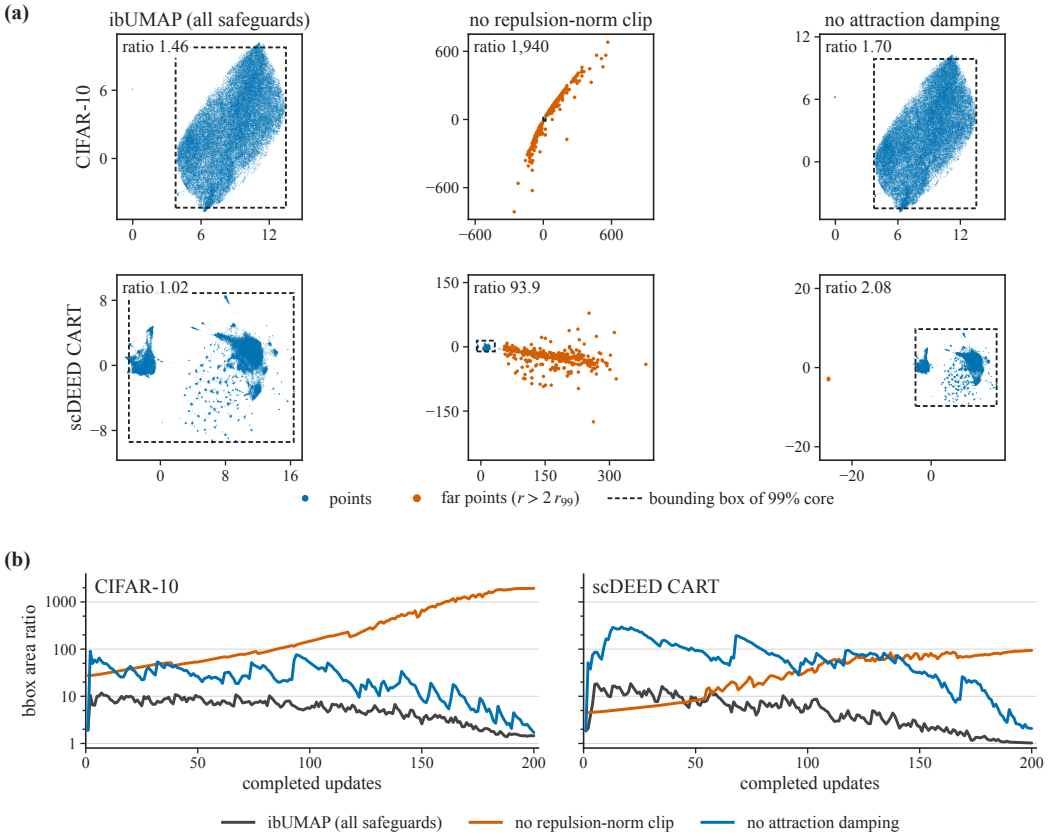


Figure 5: Safeguard failure cases. **(a)** Final embeddings of production ibUMAP and of the same optimizer without the repulsion-norm clip or without attraction damping. Each panel spans the full extent of its embedding at equal aspect; orange marks far points, the dashed rectangle bounds the 99% core, and the inset gives the bounding-box area ratio. **(b)** The same ratio after every update. Without the clip, escaped points persist to the end; without damping, displaced hubs dominate the layout for most of the run and return only as the learning rate decays.

Without the repulsion-norm clip, points escape at the first update, which is purely repulsive and uses the full learning rate (Appendix A). Its largest step is 897 on CIFAR-10 and 418 on scDEED CART, whereas the clip limits it to $4\alpha_0 = 4$. The escaped points are mostly graph hubs, with a median weighted-degree percentile above 99, consistent with the degree weight E_i in equation 2. They keep

drifting outwards, so 555 of 60,000 and 483 of 62,167 points remain far in the final layouts, and the ratio reaches 1,940 and 93.9: the main body fills only a corner of a default scatter plot (Figure 5a).

Without attraction damping, the failure appears during optimization rather than in the final layout. Attraction accumulated over many edges reaches norm 455 at a scDEED CART hub, against 102 with damping, and throws hubs out of the main body: the ratio exceeds 10 after 175 of the 200 updates and peaks at 296 (Figure 5b). While the hubs are displaced, the adaptive mesh must also cover them, and the longer side of the 99% core spans a median of 28.5 mesh boxes over the run, against 61.3 with damping. As the learning rate decays, the hubs are pulled back. Only the vertex of largest weighted degree (1,999, against a 99th percentile of 45.7) remains far, and the final ratio of 2.08 still exceeds that of every H and UMAP layout in the collection. CIFAR-10 behaves alike, with a ratio above 10 after 157 updates and a final ratio of 1.70. Damping thus keeps the layout from depending on this late recovery; smaller transient excursions remain with both safeguards, with peak ratios of 11.8 and 18.6.

These cases motivate the safeguards but do not calibrate them. The thresholds, $4\alpha_t$ for the clip and the 99th percentile with exponent 0.5 for damping, were fixed while developing an earlier version of the optimizer, and we have not repeated that selection for the final version.

Interpolation order. The default ibFFT setting uses $p = 1$. A separate optimization-only benchmark (30 datasets from six families, 150–129,062 observations, fixed fuzzy graph and spectral initialization, boundary `optimize_from_graph`) compares fixed $p = 1$, $p = 2$, and $p = 3$ on CPU and CUDA. Table 6 reports paired per-dataset differences against $p = 1$. Median changes do not exceed 4×10^{-4} on any of the three local metrics and the improvement counts stay near half the datasets, while the CPU optimizer time rises by 10.3% at $p = 2$ and 23.9% at $p = 3$. A persistent $p1 \to p2 \to p3$ stage schedule behaves likewise. Within this 30-dataset sweep, increasing the interpolation order offers no consistent improvement in these three local metrics. Together with the small average E→F differences in Table 3, this suggests limited final-quality benefit from higher interpolation orders in the tested configurations, but does not establish the numerical accuracy of the approximate field. Three caveats apply. This benchmark predates the reported implementation and its metric set differs from the main study, which adds RTA and ρ_D; its three repeats produce bitwise-identical embeddings for every deterministic variant, so each dataset contributes one quality record and three timing replicates; and $p > 1$ also forgoes the $p = 1$ particle-to-mesh path and enlarges the padded mesh to $M = (2pB)^2$, so the runtime ratios combine interpolation arithmetic, mesh growth, and deposition mode.

Table 6: Interpolation-order sweep on 30 datasets, paired against $p = 1$. Each quality cell gives the median signed difference and, in parentheses, the number of datasets improved. The last column is the median ratio of optimizer wall time to $p = 1$; values above one indicate a slowdown.

Device	Order	TW	C	NP	Time
CPU	$p = 2$	$+0.00000$ (16/30)	$+0.00001$ (18/30)	$+0.00025$ (16/30)	$1.103\times$
	$p = 3$	-0.00005 (13/30)	$+0.00002$ (19/30)	-0.00013 (14/30)	$1.239\times$
CUDA	$p = 2$	-0.00009 (11/30)	$+0.00002$ (21/30)	-0.00036 (13/30)	$1.012\times$
	$p = 3$	-0.00003 (13/30)	$+0.00001$ (20/30)	-0.00003 (14/30)	$1.019\times$

Table 7: Family-weighting sensitivity check. Entries average dataset scores within each of the five families, then average the families equally. Variant IDs and metric directions match Table 3; deterministic seed reuse is handled identically.

Variant	TW ↑	C ↑	NP ↑	RTA ↑	ρ_D ↑
A	0.9541	0.9758	0.3413	0.6989	0.5420
B	0.9460	0.9728	0.3182	0.6941	0.5213
C	0.9485	0.9734	0.3327	0.6940	0.5202
D	0.9377	0.9149	0.3194	0.6405	0.3401
E	0.9416	0.9683	0.2968	0.6956	0.5269
F	0.9421	0.9683	0.2978	0.6960	0.5258
G	0.9417	0.9758	0.2968	0.7009	0.5445
H	0.9504	0.9761	0.3274	0.6971	0.5307

Under equal family weights, synchrony's NP change remains negative (-0.0231), the FFT change remains small ($+0.0010$), and production ibUMAP remains below UMAP on NP (-0.0139) and both global scores. The kernel cap still raises mean C while lowering mean NP under family weighting. Family weighting limits dominance by numerous related subsets but does not remove all dependence, and the image family remains limited to four datasets. The fixed spectral initialization also leaves generalization to other initializations untested.

C End-to-End Benchmark: Protocol and Additional Results

C.1 Protocol, implementation profiles, and data coverage

All fits in Section 5 follow the frozen end-to-end benchmark protocol `20260806.1`, published on 9 August 2026. It contains 3,400 successful runs and 10,200 valid local-fidelity records. We additionally evaluate RTA and ρ_D on all 1,700 saved unseeded embeddings and the first seeded repeat of each dataset–method pair (340 embeddings), adding 4,080 global-fidelity records without refitting. All scheduled fits completed successfully. The nominal grid is 71 datasets, 5 implementations, 2 execution profiles, and 5 repetitions. The implementations are umap-learn, ibUMAP CPU, cuML UMAP, ibUMAP CUDA, and TorchDR UMAP. The 150 resource-skipped combinations account for the difference between the nominal 3,550 runs and the 3,400 executed runs.

Baseline scope. FastUMAP accelerates the upstream representation through a point–landmark fuzzy graph and a Nyström spectral warm start (Li, 2026), changing both the graph representation and the initialization. We exclude it because its preprocessing, threading, landmark ratio, timing boundary, and quality evaluation cannot be matched independently of those upstream changes.

Coverage and exclusions. The task builder excludes both profiles of umap-learn, ibUMAP CPU, and TorchDR on GIST, Google News, and MERFISH animals 1, 2, and 3. These exclusions were configured because of known host or device memory exhaustion. They are *resource skips*, not observed failures among the scheduled v3 runs, and do not establish a memory-capacity advantage from a controlled memory benchmark. cuML and ibUMAP CUDA both cover all 71 datasets. CPU and TorchDR comparisons cover 66, reaching 1,136,218 observations. Table 17 lists every dataset, including subsets derived from the same underlying collection.

Table 1 uses the intersection of datasets with completed runs for all five implementations: the 66 datasets outside these five exclusions. The same set is available under both execution profiles. This common set supports a five-method quality comparison without repeating ibUMAP CUDA under different populations. It is determined by coverage, not selected by quality outcomes. Runtime and stability comparisons retain the full available pairs: 66 datasets versus umap-learn, 71 versus cuML, and 66 versus TorchDR. The additional five datasets remain in the complete GPU quality summaries below. In the manifest, daggers mark datasets outside the common set.

Inputs and common settings. Algorithms receive the same processed feature file for a dataset, verified by its SHA-256 hash and loaded as `float32`. The stored inputs have dataset-specific

preprocessing; the benchmark does not replace these inputs with a common new preprocessing pipeline. Each method builds its own graph and initializes its own embedding through its full fitting path. The common requested parameters are two output dimensions, Euclidean distance, 15 neighbors, 200 epochs, `min_dist=0.1`, spread 1, learning rate 1, repulsion strength 1, negative-sample rate 5, fuzzy-set mixing ratio 1, and local connectivity 1. Low-memory mode and `n_jobs=-1` are requested where supported. The adapters record effective and unsupported parameters. In particular, TorchDR maps the epoch budget to 200 iterations and does not consume the requested local-connectivity, low-memory, thread-count, or fuzzy-set-mixing parameters. Equal requested budgets do not imply equal update counts, matched convergence, or identical graphs and starts.

ibFFT grid defaults. Both ibUMAP paths use interpolation order $p = 1$, kernel cap $c_K = 4$, and $\varepsilon = 10^{-3}$. At each epoch the mesh spans the square hull of the current coordinates, and the requested box count per dimension is $s_B \min\{4\sqrt{n},\, \max[\, 2\sqrt{n/\log n},\, B_{\min},\, R/\delta\,]\}$, where R is the coordinate range, $s_B = 1$ is the box-scale multiplier, $B_{\min} = 100$, and $\delta = 1$ intervals per unit. The request is then raised to the smallest supported FFT size strictly above it and capped at 6912, so the effective minimum is 108 boxes per dimension. The box width is rounded upward onto a logarithmic scale with 64 levels per octave, which preserves coverage and lets nearby epochs reuse a cached kernel spectrum. Near-field exactness corrections and kernel sub-sampling are available in the implementation but disabled in every reported run.

Execution profiles. The seeded umap-learn and cuML baselines pass `random_state=42`; the unseeded baselines omit it. cuML's graph-routing options and `force_serial_epochs` are left to the installed library defaults. Seeded umap-learn resolves to one job with nonparallel optimization, whereas its unseeded profile permits parallel execution. Seeded ibUMAP sets `deterministic=true`; its CPU graph policy resolves to eight jobs, and its CUDA deposition mode is segmented. Unseeded ibUMAP disables deterministic mode; the CPU graph policy requests all available jobs, and CUDA deposition is atomic. Both recorded CPU profiles use serial P2M deposition. The seed-cost comparison therefore includes these execution policies, rather than isolating the effect of a random-number seed alone.

TorchDR uses FAISS on CUDA with distributed execution and compilation disabled. Its seeded profile sets seed 42, deterministic PyTorch/cuDNN algorithms, and the deterministic cuBLAS workspace configuration `:4096:8`. Its unseeded profile disables deterministic algorithms and uses the recorded fast-mode settings. The TorchDR runs use a separate package environment on the same recorded hardware configuration.

Table 8: Recorded benchmark environment. Values describe the execution host, not the host used to prepare this manuscript.

Component	Recorded value
GPU	NVIDIA RTX A5000, 24 GiB class
Operating system	Linux 5.15.0-124, x86_64, glibc 2.31
Python	3.11.15
CPU	Intel Core i7-11700K, 8 cores / 16 hardware threads, 3.60 GHz nominal frequency
Host RAM	32 GB class (`MemTotal` 31.2 GiB)
umap-learn / PyNNDescent	0.5.12 / 0.5.13
cuML / cuVS / RMM	26.6.0 / 26.6.0 / 26.6.0
CuPy / pyFFTW	13.6.0 / 0.15.1
NumPy / SciPy / Numba	1.26.4 / 1.16.3 / 0.61.2
TorchDR / PyTorch	0.4 / 2.12.1+cu126

The exported hardware signature is shared across the runs and includes the operating-system platform and GPU description. The export records neither CPU model nor host memory; both are supplemented from the execution host, the CPU from `lscpu` and the memory from `/proc/meminfo` as captured in the environment record of the downstream experiment (Appendix D), which ran on the same host with the same kernel, Python build, and CPU. The results characterize the recorded execution configuration rather than hardware-independent performance. Per-family runtime and source

signatures prevent incompatible results from being silently combined. This appendix uses the accepted frozen bundle, rather than revalidating its numerical claims against a later source checkout.

Timing and repetitions. Each run uses a fresh worker process. After loading and validating the input, the worker performs one warmup of its configured algorithm before measuring the fitting call. The measured region includes construction, graph building, initialization, optimization, synchronization, and transfer of the result to host memory. It excludes worker startup, input loading, hashing, warmup, output serialization, evaluation, and plotting. Runs are shuffled within repetition blocks. This is warm-call end-to-end timing, not cold-start latency. The primary field is `e2e_call_wall_s`; the export's legacy `optimization_call_wall_s` field is an alias of the same end-to-end measurement and is not analyzed as a separate timing boundary. Resource snapshots do not provide a comparable peak-device-memory measurement, so no such comparison is inferred here.

Aggregation. Let $t_{d,a,r}$ denote the recorded runtime on dataset d, algorithm/profile a, and repetition r. We use $\widetilde{t}_{d,a} = \mathrm{median}_{r=1}^{5}\, t_{d,a,r}$ and the paired speed ratio $s_d = \widetilde{t}_{d,\mathrm{baseline}}/\widetilde{t}_{d,\mathrm{ibUMAP}}$. A ratio above one favors ibUMAP. Suite and size-group summaries take the median of these dataset-level ratios. Main-text fidelity uses five-run mean scores within each dataset, then takes medians across the common 66 datasets. Appendix C.3 gives the seeded quality-score convention and distinguishes means, marginal medians, and paired-difference medians. Repeated timings or identical-score reuse records are not treated as additional datasets. Size bins and collection weighting are descriptive summaries of this fixed suite, not a random sample of all potential datasets or a pre-registered non-inferiority study.

C.2 Additional end-to-end performance results

Table 9 separates the CPU fixed-overhead regime from larger datasets and retains both profiles for the GPU comparison. CPU ibUMAP loses on all six datasets below 1,000 observations and wins on every tested larger dataset. The suite has no observations between 646 and 1,567 rows, and the datasets also differ in dimensionality and structure. The first dataset showing a speedup therefore does not establish a controlled CPU crossover.

Table 9: End-to-end speed ratios, baseline/ibUMAP, summarized by dataset size. S: seeded; U: unseeded. "Wins" counts dataset-level ratios strictly greater than one. CPU/UMAP compares ibUMAP CPU with umap-learn; both GPU comparisons use ibUMAP CUDA.

Comparison	Size group	n	Median S	Median U	Wins S	Wins U
CPU/UMAP	All	66	5.792	3.286	60/66	60/66
CPU/UMAP	$N < 10^3$	6	0.310	0.167	0/6	0/6
CPU/UMAP	$10^3 \le N < 10^4$	13	5.724	3.832	13/13	13/13
CPU/UMAP	$10^4 \le N < 10^5$	41	5.937	3.042	41/41	41/41
CPU/UMAP	$10^5 \le N < 10^6$	5	7.310	4.077	5/5	5/5
CPU/UMAP	$N \ge 10^6$	1	5.253	4.466	1/1	1/1
GPU/cuML	All	71	0.726	0.875	17/71	25/71
GPU/cuML	$N < 10^4$	19	0.253	0.359	2/19	3/19
GPU/cuML	$10^4 \le N < 10^5$	41	0.726	0.905	7/41	11/41
GPU/cuML	$10^5 \le N < 10^6$	5	0.990	1.062	2/5	5/5
GPU/cuML	$N \ge 10^6$	6	1.045	1.444	6/6	6/6
GPU/TorchDR	All	66	1.388	1.532	60/66	62/66

Against cuML, the unseeded median speed ratio rises from 0.359 below 10^4 observations to 0.905 in the 10^4–10^5 bin, 1.062 in the 10^5–10^6 bin, and 1.444 at or above 10^6. Seeded ibUMAP also wins on all six million-scale datasets, but its median advantage is only $1.05\times$: modest, not evidence of a large seeded GPU speedup. Figure 6 retains all individual paired speed ratios.

The CPU repeatability result concerns the absence of an aggregate runtime penalty: ibUMAP's median ratio is 0.961, but 4/66 datasets have a seeded median runtime more than 10% above their unseeded median. Figure 7 retains all dataset-level ratios underlying the CPU repeatability comparison in Section 5.2.

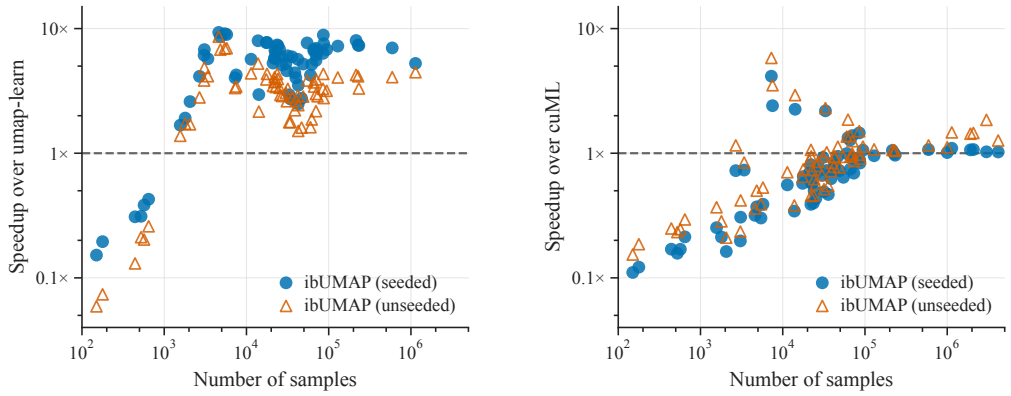


Figure 6: Dataset-level end-to-end speed ratios for CPU (left) and GPU versus cuML (right). Each ratio compares matching seeded or unseeded profiles after taking each algorithm's five-run median time. The horizontal line denotes equal runtime; values above it favor ibUMAP. Both axes are logarithmic. CPU and GPU contain 66 and 71 paired datasets per profile, respectively.

Table 10: Runtime cost and observed output identity of the seeded profile. Time ratios are seeded/unseeded, so values above one indicate a seeded runtime penalty. "Identical" counts datasets with identical full embedding hashes in all five seeded runs. The million-scale column has only one dataset for CPU and TorchDR.

Implementation	n	Median ratio	$n_{\geq 10^6}$	Ratio $N \geq 10^6$	Identical
umap-learn	66	1.664	1	1.170	66/66
ibUMAP CPU	66	0.961	1	0.995	66/66
cuML	71	1.071	6	9.191	70/71
ibUMAP CUDA	71	1.306	6	11.718	71/71
TorchDR	66	1.136	1	1.044	66/66

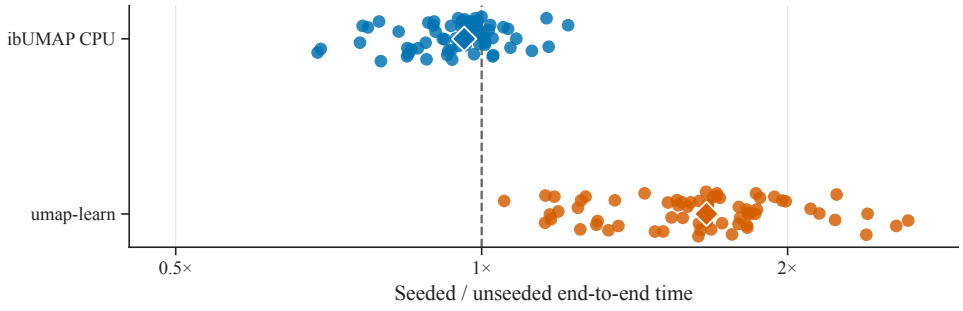


Figure 7: CPU runtime cost of seeded execution. Each point is one dataset's seeded five-run median time divided by its unseeded five-run median, with 66 datasets per implementation. Diamonds mark medians across datasets. The horizontal scale is logarithmic; ratios above one indicate a seeded runtime penalty. Vertical jitter separates points.

GPU behavior differs substantially. Across the six million-scale datasets, seeded/unseeded ratios have medians 11.718 for ibUMAP CUDA and 9.191 for cuML. The ibUMAP ratios range from 4.360 to 48.215. Figure 8 gives the absolute CPU and GPU runtimes for both profiles. The unseeded GPU speedups in the main text must not be interpreted as the runtime of the seeded configuration. Figure 9 attributes this seeded cost almost entirely to graph construction.

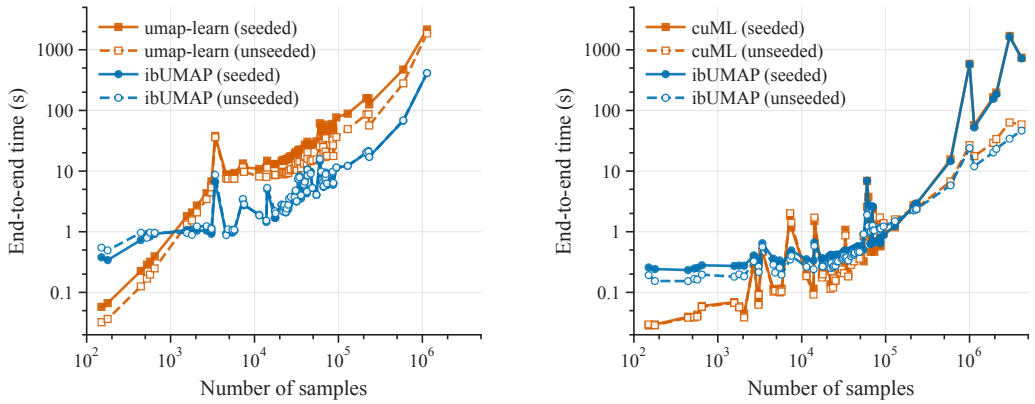


Figure 8: Absolute end-to-end runtime for both execution profiles: CPU versus umap-learn (left, 66 datasets) and GPU versus cuML (right, 71 datasets). Each point is a five-run median. Both axes are logarithmic. Connecting lines are visual guides across different datasets, not controlled size sweeps. Seeded and unseeded fitting paths can have markedly different costs on large GPU workloads.

The main-text TorchDR comparison uses its 66 datasets with completed unseeded runs. Its seeded counterpart gives ibUMAP CUDA a median speedup of 1.39×, with 60/66 wins, compared with 1.53× and 62/66 unseeded. These comparisons do not include the five resource-skipped datasets and are not substituted for the 71-dataset cuML comparison.

Stage composition. Figure 9 divides each ibUMAP end-to-end time into the stages reported by its internal timers: graph construction, other preprocessing, spectral initialization, optimization, and the unattributed remainder of the measured call. Shares use five-run medians per dataset; absolute times are in Figure 8. Below 10^4 observations, optimization takes a median 91% of unseeded CPU time and 67% of unseeded CUDA time. From 10^4 observations onward, initialization becomes the largest CPU stage, with median shares of 43% in the 10^4–10^5 bin and 54% at or above 10^5; optimization falls to 22% in the latter bin. On CUDA, graph construction dominates at or above 10^5 observations, with median shares of 80% unseeded and 96% seeded, while optimization takes 8% and 2%. The seeded GPU penalty on the six million-scale datasets therefore arises in graph construction: on Google News, for example, graph construction takes 1,638 s seeded versus 27 s unseeded, whereas optimization changes only from 3.0 s to 3.7 s. The cause is the neighbor search: GPU NN-Descent does not currently guarantee repeatable graphs, so the seeded profile always builds the k-NN graph by brute force, whereas the unseeded profile switches to NN-Descent above 50,000 observations. The isolated peaks in the graph share at smaller sizes correspond to high-dimensional inputs ($D \geq 784$, against a suite median of 50). The unattributed remainder has a median share of at most 2% in every size bin, and no run's recorded stages exceed its measured time. The baselines do not expose comparable stage timers, so this decomposition does not support stage-level comparisons across implementations.

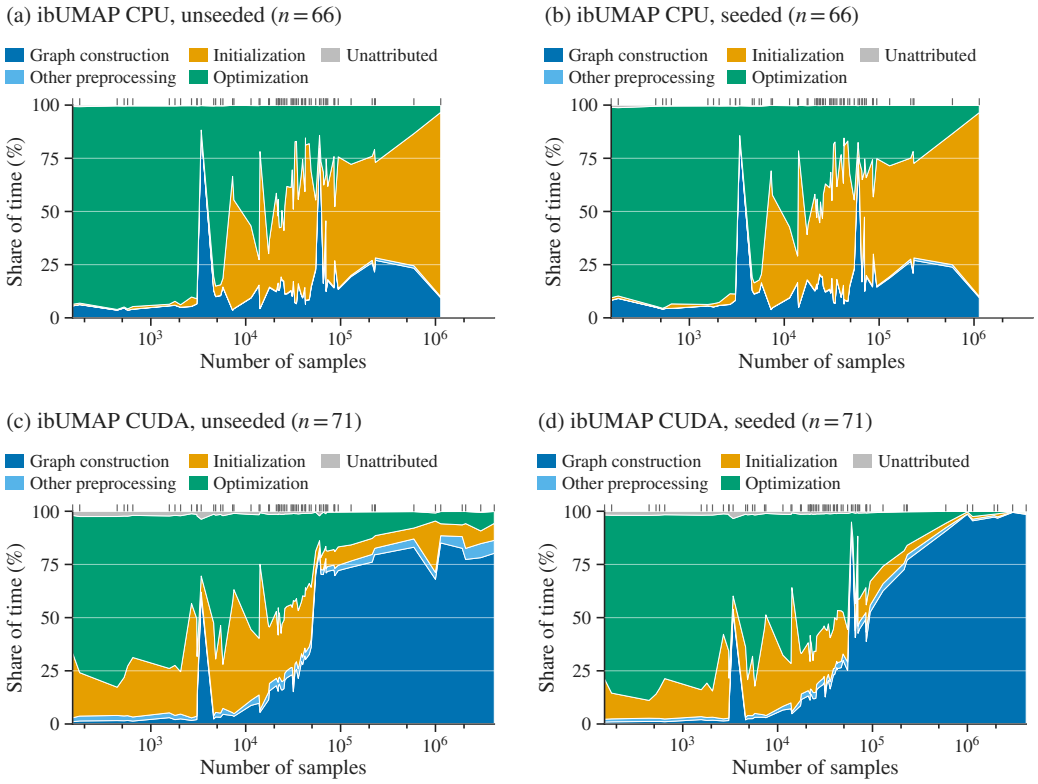


Figure 9: Shares of ibUMAP end-to-end time by stage: CPU (a, b; 66 datasets) and CUDA (c, d; 71 datasets) under both execution profiles. For each dataset, every stage's five-run median time is divided by the sum of these medians. Unattributed time is the measured call time not covered by the recorded stages. Tick marks along the top edge mark dataset sizes. Areas between them are visual interpolations across different datasets, not controlled size sweeps, and the two 70,000-observation datasets are averaged. The horizontal axis is logarithmic and shared by all panels.

C.3 FIDELITY METRICS AND ADDITIONAL COMPARISONS

C.3.1 EVALUATION MEASURES AND AGGREGATION

All five fidelity metrics use the processed feature representation used for fitting. Evaluation uses all rows when $n \leq 10^6$, otherwise a uniform sample of 10^6 rows without replacement. All five metrics use sampling seed 42 and the same evaluation indices across methods and runs. Local scores describe neighborhoods of the induced evaluation set. The full embedding is fitted before sampling; identical embedding hashes permit score reuse while preserving each run's provenance.

Local fidelity measures. Trustworthiness penalizes embedding-space neighbors that are distant in input-space rank; continuity penalizes input-space neighbors that are distant in embedding-space rank (Venna & Kaski, 2006; Lee & Verleysen, 2009). Neighborhood preservation reports the mean fraction of shared neighbors between the input and embedding neighbor sets (Huang et al., 2022). All use $k = 15$, exclude self-neighbors, and are higher-is-better scores.

Let $N_X^k(i)$ and $N_Y^k(i)$ denote the k nearest neighbors of observation i in the input and embedding spaces, and let $r_X(i,j)$ and $r_Y(i,j)$ be the corresponding one-based neighbor ranks. With $U_i = N_Y^k(i) \setminus N_X^k(i)$, $V_i = N_X^k(i) \setminus N_Y^k(i)$, and $Z = 2/[nk(2n - 3k - 1)]$, the scores are

$$\mathrm{TW}(k) = 1 - Z \sum_{i=1}^{n} \sum_{j \in U_i} \big(r_X(i,j) - k\big), \tag{7}$$

$$\mathrm{C}(k) = 1 - Z \sum_{i=1}^{n} \sum_{j \in V_i} \big(r_Y(i,j) - k\big), \tag{8}$$

$$\mathrm{NP}(k) = \frac{1}{nk} \sum_{i=1}^{n} \left| N_X^k(i) \cap N_Y^k(i) \right|. \tag{9}$$

TW and C weight rank errors, whereas NP counts retained neighbors without rank weighting. These measures characterize local neighborhood fidelity.

Global fidelity measures. RTA is the fraction of sampled triplets (i, j, k) preserving whether j or k is closer to anchor i; ρ_D is the Spearman correlation of input and embedding distances over sampled point pairs (Huang et al., 2022). Both use Euclidean distances on the stored `float32` arrays. We draw 10^6 ordered triplets with replacement, with distinct indices within each triplet, exclude input-distance ties, and count embedding-distance ties as incorrect. For m evaluation points, we sample $\min\{10^6, m(m-1)/2\}$ distinct unordered pairs without replacement, using all pairs when fewer are available; Spearman uses average ranks for ties. Triplets and pairs use seed 42 and are shared across methods and runs. RTA lies in $[0, 1]$ and ρ_D in $[-1, 1]$, with higher values preferred. Unlike the stability measure below, ρ_D compares inputs with embeddings, not two reruns. These global scores cover all nine scored mechanism variants and both end-to-end profiles on their full available dataset sets.

The seeded profile is intended to reproduce the same embedding on the same device, so all five quality metrics use its first run; additional runs measure runtime and check output identity (Appendix C.4). Unseeded quality uses the mean of five runs. For metric q, let $\bar{q}_{d,a}$ denote this dataset-level score for implementation/profile a. On a specified dataset set D, the absolute mean and marginal median are $\mathrm{mean}_{d \in D}\, \bar{q}_{d,a}$ and $\mathrm{median}_{d \in D}\, \bar{q}_{d,a}$. The main table uses the latter on the common 66 datasets. A paired difference is instead $\Delta_d = \bar{q}_{d,\mathrm{ibUMAP}} - \bar{q}_{d,\mathrm{baseline}}$. Its mean equals the difference of means on a shared set, whereas its median need not equal the difference of marginal medians. Neither summary counts per-dataset wins or establishes statistical significance. No composite quality score or post hoc noninferiority tolerance is introduced.

C.3.2 EXTENDED FIDELITY SUMMARIES

Tables 11 and 12 cover both profiles on the common 66 datasets and the full 71-dataset cuML–ibUMAP CUDA pair. Bold marks the best score within each coverage block and profile.

Table 11: Mean fidelity by profile and coverage. Dataset scores use the first seeded run or the five-run unseeded mean; datasets receive equal weight. Bold marks the best score within each block and profile.

Method	Profile	TW ↑	C ↑	NP ↑	RTA ↑	ρ_D ↑
CPU: common 66 datasets						
umap-learn	Unseeded	**0.9728**	**0.985230**	**0.3983**	**0.7254**	**0.5952**
ibUMAP	Unseeded	0.9672	0.984801	0.3762	0.6961	0.5292
umap-learn	Seeded	**0.9727**	**0.985119**	**0.3979**	**0.7286**	**0.6006**
ibUMAP	Seeded	0.9673	0.984823	0.3758	0.6971	0.5306
GPU: common 66 datasets						
cuML	Unseeded	**0.9715**	0.985300	0.3927	0.7299	0.6037
TorchDR	Unseeded	0.9710	0.984591	**0.4028**	**0.7532**	**0.6435**
ibUMAP	Unseeded	0.9671	**0.985545**	0.3758	0.7139	0.5640
cuML	Seeded	0.9680	0.985258	0.3812	0.7263	0.5939
TorchDR	Seeded	**0.9712**	0.984450	**0.4030**	**0.7529**	**0.6424**
ibUMAP	Seeded	0.9671	**0.985612**	0.3761	0.7108	0.5553
GPU: full 71 datasets						
cuML	Unseeded	**0.9652**	**0.982812**	**0.3664**	**0.7243**	**0.5912**
ibUMAP	Unseeded	0.9602	0.982737	0.3506	0.7080	0.5504
cuML	Seeded	**0.9623**	0.983954	**0.3557**	**0.7202**	**0.5795**
ibUMAP	Seeded	0.9605	**0.984558**	0.3509	0.7054	0.5422

Table 12: Median fidelity for the same blocks and profiles as Table 11, using the first seeded run or the five-run unseeded mean per dataset. Common-set unseeded rows reproduce Table 1 with extra precision for C. Bold marks the best score before rounding.

Method	Profile	TW ↑	C ↑	NP ↑	RTA ↑	ρ_D ↑
CPU: common 66 datasets						
umap-learn	Unseeded	**0.9901**	0.994200	**0.3699**	**0.7249**	**0.6072**
ibUMAP	Unseeded	0.9853	**0.995031**	0.3436	0.7023	0.5493
umap-learn	Seeded	**0.9902**	0.994569	**0.3677**	**0.7259**	**0.6031**
ibUMAP	Seeded	0.9852	**0.995017**	0.3443	0.6995	0.5396
GPU: common 66 datasets						
cuML	Unseeded	**0.9892**	0.994996	0.3635	0.7349	0.6295
TorchDR	Unseeded	0.9873	0.994347	**0.3768**	**0.7647**	**0.6706**
ibUMAP	Unseeded	0.9851	**0.995005**	0.3407	0.7217	0.5897
cuML	Seeded	0.9864	**0.995175**	0.3502	0.7288	0.6076
TorchDR	Seeded	**0.9873**	0.994281	**0.3776**	**0.7648**	**0.6692**
ibUMAP	Seeded	0.9852	0.995017	0.3413	0.7179	0.5789
GPU: full 71 datasets						
cuML	Unseeded	**0.9878**	**0.993971**	**0.3414**	**0.7266**	**0.6061**
ibUMAP	Unseeded	0.9841	0.993892	0.3267	0.7156	0.5725
cuML	Seeded	**0.9864**	**0.994454**	**0.3349**	**0.7221**	**0.5939**
ibUMAP	Seeded	0.9844	0.993905	0.3329	0.7140	0.5500

Table 13 summarizes differences within each baseline pair using its full available dataset set (66/71/66). It retains both the mean and the median of paired differences, separately from the marginal score medians above.

Table 13: Means and medians of paired fidelity differences, ibUMAP minus baseline, on each pair's full available set. Dataset-level scores are paired before aggregation: the first seeded run or the mean of five unseeded runs. Positive differences favor ibUMAP for all five metrics. C retains six decimals in the median panel to avoid rounding a small negative difference to zero.

Comparison	Profile	n	ΔTW	ΔC	ΔNP	ΔRTA	$\Delta\rho_D$
Mean of paired differences							
CPU/UMAP	Seeded	66	-0.0054	-0.0003	-0.0221	-0.0315	-0.0700
CPU/UMAP	Unseeded	66	-0.0056	-0.0004	-0.0221	-0.0293	-0.0660
GPU/cuML	Seeded	71	-0.0018	$+0.0006$	-0.0048	-0.0148	-0.0372
GPU/cuML	Unseeded	71	-0.0051	-0.0001	-0.0158	-0.0162	-0.0408
GPU/TorchDR	Seeded	66	-0.0041	$+0.0012$	-0.0269	-0.0421	-0.0870
GPU/TorchDR	Unseeded	66	-0.0039	$+0.0010$	-0.0270	-0.0393	-0.0795
Median of paired differences							
CPU/UMAP	Seeded	66	-0.0028	$+0.000094$	-0.0213	-0.0191	-0.0541
CPU/UMAP	Unseeded	66	-0.0030	-0.000034	-0.0222	-0.0155	-0.0522
GPU/cuML	Seeded	71	-0.0010	-0.000126	-0.0038	-0.0049	-0.0210
GPU/cuML	Unseeded	71	-0.0031	-0.000051	-0.0163	-0.0106	-0.0257
GPU/TorchDR	Seeded	66	-0.0025	$+0.000483$	-0.0304	-0.0307	-0.0549
GPU/TorchDR	Unseeded	66	-0.0021	$+0.000587$	-0.0297	-0.0303	-0.0532

C.3.3 Dataset-level variation and aggregation sensitivity

Figure 10 retains all individual unseeded differences in the five fidelity metrics on the full paired sets. Unlike the common-set main table, its cuML comparison includes all 71 datasets. The spread shows why a single mean or median cannot characterize every dataset.

Summary choices can change the apparent ordering. On the common 66 datasets, CPU ibUMAP has a higher continuity marginal median than umap-learn, but its mean difference is -0.000429 and its paired-difference median is -0.000034. The higher marginal median therefore does not demonstrate an improvement on a typical paired dataset. Reporting both summaries exposes this change instead of treating the most favorable summary as a uniform quality advantage. Small continuity differences also do not cancel the larger changes in neighborhood preservation.

Collection weighting. The suite contains 35 Tabula Sapiens entries, 11 C. elegans entries, and 4 MERFISH entries, alongside 21 other datasets. Several entries represent related or overlapping subsets. As a post hoc aggregation check, we average the paired score differences within each of these three collections, retain each remaining dataset as a singleton, and then weight the resulting blocks equally. This gives 24 blocks for the cuML comparison and 22 for CPU and TorchDR after resource skips. It reduces the multiplicity of these three collections; it does not establish independent biological replicates or a uniquely correct population weighting.

Table 14: Collection-weighted mean fidelity differences on the full paired sets, using the explicit grouping above. Positive differences favor ibUMAP. The corresponding dataset-weighted means are in the mean panel of Table 13.

Comparison	Profile	Blocks	ΔTW	ΔC	ΔNP	ΔRTA	$\Delta\rho_D$
CPU/UMAP	Seeded	22	-0.0029	-0.0011	-0.0083	-0.0361	-0.0717
CPU/UMAP	Unseeded	22	-0.0029	-0.0016	-0.0077	-0.0290	-0.0608
GPU/cuML	Seeded	24	-0.0015	$+0.0021$	$+0.0012$	-0.0014	$+0.0024$
GPU/cuML	Unseeded	24	-0.0044	-0.0002	-0.0041	-0.0092	-0.0192
GPU/TorchDR	Seeded	22	-0.0071	$+0.0023$	-0.0195	-0.0268	-0.0496
GPU/TorchDR	Unseeded	22	-0.0068	$+0.0017$	-0.0195	-0.0274	-0.0500

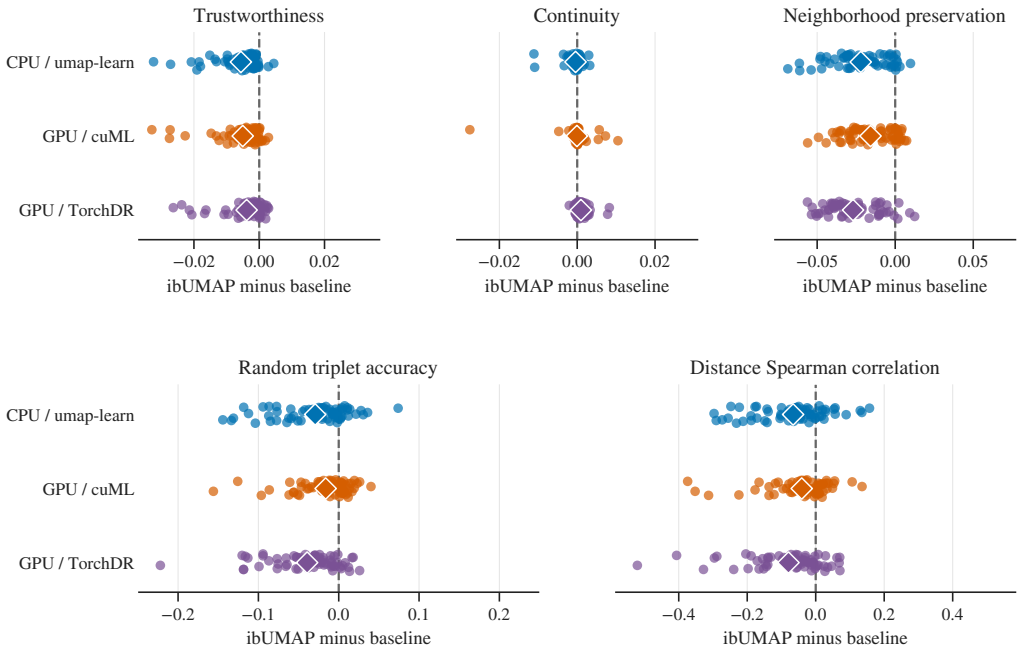


Figure 10: Unseeded fidelity differences on the full available pairs: CPU/umap-learn (66 datasets), GPU/cuML (71), and GPU/TorchDR (66). Each point is ibUMAP's five-run mean score minus the baseline's; diamonds mark means across datasets. Positive differences favor ibUMAP. Horizontal scales differ between metrics and are shared across comparisons within each panel. Vertical jitter separates overlapping points.

All three full-coverage unseeded comparisons retain negative mean differences in TW, NP, RTA, and ρ_D after this reweighting. Their magnitudes depend on the weighting: for example, the trustworthiness deficit versus TorchDR increases (-0.0039 to -0.0068). Weighting can change other conclusions: seeded GPU differences in NP and ρ_D versus cuML become slightly positive. We therefore keep the grouping explicit and retain the complete dataset-level inputs instead of selecting datasets by the size or direction of their observed effects.

C.4 Repeatability and run-to-run stability protocol

Seeded output identity. For each dataset and implementation, the five complete saved embedding files are compared by SHA-256 hash. All seeded ibUMAP CPU, ibUMAP CUDA, umap-learn, and TorchDR groups are identical. Seeded cuML is identical on 70/71 datasets. On Google News, its first run differs from runs 2–5, which are identical to one another, despite using the same device, configuration, and `random_state=42`. The embedding arrays and their local quality scores differ, confirming that the discrepancy is not merely in file serialization. Thus, setting cuML's random seed did not ensure bitwise repeatability in every tested case under the recorded library defaults. The benchmark rebuilds the graph and initialization on each call, so this concerns full-pipeline repeatability. No unseeded dataset/implementation group has identical hashes across all five runs.

Geometric comparisons. For each dataset, implementation, and profile, repetition 1 is compared with repetitions 2–5, giving four comparisons per group, rather than all ten unordered pairs. A deterministic comparison-specific seed selects at most 10,000 point indices without replacement; corresponding points are shared between the two embeddings in that comparison. Different algorithms and repetition pairs can use different sampled indices. Local stability is the mean fraction of shared neighbors between their 15-nearest-neighbor sets within this sampled point set.

For geometric alignment, each sampled embedding is centered and divided by its Frobenius norm. An orthogonal Procrustes fit, allowing reflection, aligns the coordinates before RMS displacement is computed. The distance-correlation measure uses randomly drawn index pairs, up to $\min(20{,}000, 4m)$ for m sampled points, discarding self-pairs. It reports Spearman correlation between distances in the two embeddings. These similarity measures are not affected by global translations, rotations, or uniform scale changes in the way raw coordinate differences would be. The Procrustes RMS depends on this normalization and should not be read as a displacement in the original embedding's units.

Each dataset's score is the median over its four reference-to-repeat comparisons. Table 15 then takes medians across datasets. Table 16 compares algorithms within each shared dataset before counting improvements and aggregating differences; this preserves the appropriate population for each baseline. Figure 3c displays the 15-NN overlap differences with their cross-dataset medians as diamonds, matching the paired medians in the table.

Table 15: Unseeded run-to-run stability. Each entry is a median across dataset-level median scores. Higher neighbor overlap and distance correlation, and lower aligned RMS, indicate greater stability.

Implementation	n	Neighbor overlap	Distance correlation	Procrustes RMS
umap-learn	66	0.6929	0.9454	0.002100
ibUMAP CPU	66	0.8369	0.9902	0.000753
cuML	71	0.7311	0.9826	0.001065
ibUMAP CUDA	71	0.8851	0.9983	0.000337
TorchDR	66	0.6851	0.9632	0.001345

Table 16: Dataset-paired unseeded stability comparisons. Differences are ibUMAP minus baseline. A difference of medians in Table 15 need not equal the median of paired differences shown here.

Comparison	n	Overlap wins	Median Δoverlap	Corr. wins	Median Δcorr.
CPU/UMAP	66	58/66	$+0.1436$	52/66	$+0.0357$
GPU/cuML	71	67/71	$+0.1308$	68/71	$+0.0147$
GPU/TorchDR	66	66/66	$+0.1961$	62/66	$+0.0256$

ibUMAP improves neighbor overlap on 58/66 datasets versus umap-learn, 67/71 versus cuML, and 66/66 versus TorchDR. Distance correlation improves on 52/66, 68/71, and 62/66, respectively. The advantage is therefore prevalent rather than universal. It describes variation across unseeded full-pipeline reruns, which can differ in graph construction, initialization, and execution order; it is not a controlled study of initialization perturbations or a causal attribution to the repulsive-field evaluator. Lower run-to-run variation also does not imply higher fidelity to the input data.

C.5 Complete dataset manifest

Table 17 preserves the complete suite, including small datasets and related subsets. TS abbreviates the `tabula_sapiens_v2` prefix; CE abbreviates `c_elegans_embryogenesis`; MERFISH abbreviates `whole_mouse_brain_merfish`. Spaces replace underscores in the displayed identifiers, and a slash separates paired cell-type names. The scDEED prefix denotes dataset provenance, not an additional scDEED score in this benchmark. D is the number of features in the processed fitting input.

Data sources. Tabula Sapiens v2 (The Tabula Sapiens Consortium, 2026); *C. elegans* embryogenesis (Packer et al., 2019) in the CaltechDATA AnnData release (da Veiga Beltrame, 2021); whole-mouse-brain MERFISH (Zhang et al., 2023); the scDEED datasets as released by Xia et al. (2024); MNIST (LeCun et al., 1998), Fashion-MNIST (Xiao et al., 2017), and CIFAR-10 (Krizhevsky, 2009); GIST-960 (Jégou et al., 2011) from ANN-Benchmarks (Aumüller et al., 2020); Google News word2vec vectors (Mikolov et al., 2013); bank, epileptic, HIVA, SECOM, seismic, and spambase as preprocessed by Espadoto et al. (2021) from their original sources (Moro et al., 2014; Andrzejak et al., 2001; Wu & Fokoue, 2017; Guyon et al., 2007; McCann & Johnston, 2008; Sikora & Wróbel, 2010; Hopkins et al., 1999); forest type (Johnson, 2012); and iris, wine, breast cancer, digits, and diabetes via scikit-learn (Pedregosa et al., 2011; Fisher, 1936; Aeberhard & Forina, 1991; Street et al., 1993; Alpaydin & Kaynak, 1998; Efron et al., 2004).

Table 17: All 71 benchmark datasets and their processed shapes. The 66 unmarked entries form the common quality set in Table 1. A dagger marks the five additional datasets evaluated only with cuML and ibUMAP CUDA; both profiles of the other implementations are resource-skipped.

Dataset	N	D
iris	150	4
wine	178	13
diabetes	442	10
foresttype	523	27
breast cancer	569	30
seismic	646	24
secom	1,567	590
digits	1,797	64
bank	2,059	63
TS: neural	2,685	50
TS: ear	3,055	50
hiva	3,076	1,617
scDEED: velocity	3,396	25,919
spambase	4,601	57
scDEED: marrow	4,821	50
CE: stage early annotated	5,442	32
epileptic	5,750	178
TS: germline	7,276	50
TS: testis	7,513	50
TS: kidney	11,376	50
CE: pair hypodermis / ciliated amphid neuron	13,836	32
TS: pancreas	14,140	50
CE: stage late annotated	17,436	32
TS: skin	17,786	50
TS: prostate	21,030	50
CE: pair body wall muscle / ciliated non amphid neuron	21,988	32
TS: uterus	22,029	50
TS: liver	22,214	50

Continued on next page

Table 17 (continued)

Dataset	N	D
TS: trachea	22,671	50
CE: pair body wall muscle / ciliated amphid neuron	23,610	32
CE: plot cell type top 8 mixture	24,095	32
CE: pair body wall muscle / hypodermis	25,266	32
TS: heart	25,832	50
TS: bone marrow	27,112	50
TS: large intestine	30,084	50
TS: mammary	30,936	50
CE: stage mid annotated	31,771	32
TS: stomach	33,064	50
TS: eye	34,273	50
CE: cell type top 4 mixture	35,824	32
TS: tongue	38,754	50
TS: salivary gland	39,821	50
TS: small intestine	42,036	50
TS: vasculature	42,650	50
TS: thymus	42,729	50
TS: muscle	46,772	50
TS: ovary	48,951	50
CE: global qc annotated	54,649	32
cifar10	60,000	3,072
scDEED: cart	62,167	1,000
TS: lung	65,847	50
TS: bladder	66,385	50
fashion mnist	70,000	784
mnist784	70,000	784
TS: spleen	70,448	50
TS: endothelium	73,195	50
TS: blood	85,233	50
CE: qc passed	86,024	32
scDEED: samusik	86,864	38
TS: fat	94,415	50
TS: lymph node	129,062	50
TS: epithelium	228,032	50
TS: stromal	232,684	50
TS: immune	592,317	50
TS: all cells	1,136,218	50
MERFISH: animal4 sagittal	215,278	50
gist 960 euclidean†	1,000,000	960
MERFISH: animal2 coronal†	1,915,592	50
MERFISH: animal3 sagittal†	2,081,549	50
google news 300d†	3,000,000	300
MERFISH: animal1 coronal†	4,167,869	50

D Downstream Reuse: BRAQUE Protocol and Additional Results

Data and preprocessing. The case uses the public L2 single-cell protein-expression sample associated with BRAQUE (Dall'Olio et al., 2023). A public antibody workbook supplies a proxy for the paper-specific feature-selection reference: we retain unambiguous normalized name matches with a nonmissing significance entry that does not contain `IGNORE`. This yields 62 features for 56,962 cells. Lognormal Shrinkage is computed once; its output is median-centered and divided by each feature's mean absolute deviation about its mean, then stored as `float32`. All repetitions use the same frozen matrix and cell order. The experiment has neither verified spatial coordinates nor final expert phenotype labels, so it evaluates computational partitions rather than spatial organization, cell-type accuracy, or expert annotation quality.

Embedding and clustering configurations. Both embedding methods use Euclidean distance, two dimensions, 50 neighbors, 200 epochs, spectral initialization, `min_dist=0`, spread 1, learning rate 1, repulsion strength 1, negative-sample rate 5, local connectivity 1, fuzzy-set mixing ratio 1, and

low-memory mode. Each fit rebuilds the graph and initialization from the frozen matrix; graphs and starting coordinates are not shared between fits. Seeded profiles set `random_state=42`; seeded ibUMAP additionally enables deterministic execution. Seeded UMAP uses one job, while unseeded UMAP and both ibUMAP profiles request eight. The process-level Numba and BLAS thread budget is eight. These are execution-profile comparisons, not an isolation of the random-number seed or optimizer alone.

HDBSCAN uses the raw fitted coordinates, Euclidean distance, `min_cluster_size=10`, `min_samples=10`, `cluster_selection_epsilon=0.1`, excess-of-mass selection, the `best` algorithm, approximate minimum-spanning-tree construction, and one core-distance job. This configuration is identical in all 20 runs. Rigid alignment is applied only for display in Figure 4, allowing rotation/reflection and translation but no scaling.

Execution and timing. The experiment runs on an Intel Core i7-11700K at 3.60 GHz with 32 GB-class RAM (`MemTotal` 31.2 GiB), Linux 5.15, Python 3.11.15, umap-learn 0.5.12, HDBSCAN 0.8.44, NumPy 1.26.4, SciPy 1.16.3, scikit-learn 1.8.0, Numba 0.61.2, PyNNDescent 0.5.13, and pyFFTW 0.15.1. Each run uses a fresh process. Four profiles are randomly interleaved within each repetition block. Within each process, embedding and clustering are warmed up on the same fixed 5,000-cell subset, after which fresh estimators fit the complete matrix. All five measured runs per profile are retained; all 20 scheduled runs succeed. The warmup does not guarantee removal of every size-dependent first-call overhead.

Timing separately surrounds `fit_transform` and `fit_predict`. Embedding time includes graph construction, initialization, and optimization, but excludes estimator construction, input loading, imports, preprocessing, warmup, serialization, and evaluation. The combined time is the sum of the two fit calls within a run, summarized by its five-run median; it is not the complete BRAQUE runtime. These boundaries differ from Section 5, which also times estimator construction and host materialization.

Evaluation and aggregation. Clustering metrics use all cells. ARI treats noise as a separate label. Assignment disagreement uses Hungarian one-to-one matching to maximize non-noise cell overlap, reserves the noise label, and gives unmatched clusters new labels. It therefore includes cluster splits, merges, and noise transitions rather than measuring error against ground truth. We summarize all ten unordered pairs within each profile by their median; these pairs share runs and are not ten independent replicates. Full coordinate and raw-label hashes test exact within-profile repeatability.

Distance correlation uses 200,000 sampled non-self cell pairs from the full fitted coordinates. Trustworthiness uses a fixed 3,000-cell induced subset; input-neighbor overlap uses a fixed 2,048-cell induced subset. Both local metrics use 15 neighbors. The same evaluation indices are used for every fit. These local scores concern subset neighborhoods, not full-data neighborhoods, and are distinct from run-to-run agreement. No noninferiority threshold or significance test is applied.

Table 18: Complete CPU reuse summary. Times are five-run medians in seconds. Distance correlation ρ, ARI, and changed-assignment fraction summarize ten unordered run pairs. "Exact" denotes identical full coordinates and raw labels across all five runs within that profile.

Profile	Embedding	+ HDBSCAN	ρ	ARI	Changed	Exact
UMAP unseeded	15.436	16.076	0.937	0.611	20.80%	No
UMAP seeded	49.062	49.696	1.000	1.000	0.00%	Yes
ibUMAP unseeded	7.095	7.722	0.999	0.601	22.01%	No
ibUMAP seeded	7.099	7.736	1.000	1.000	0.00%	Yes

For the two plotted runs, 16.29% of cells change between non-noise clusters and 3.49% change noise status, totaling 19.78%; ARI restricted to cells assigned to clusters in both runs is 0.656. The 20.80% in Table 18 is the median over all ten UMAP pairs, not the plotted pair's value. Seeded trustworthiness is 0.970230 for UMAP and 0.962791 for ibUMAP, a difference of -0.007440; input-neighbor overlap is 0.402051 and 0.370703, a difference of -0.031348. The measured benefit is lower-cost within-profile repeatability, with these fidelity differences retained. It does not imply matching

partitions across methods, improved unseeded clustering stability, cross-device reproducibility, or robustness to other seeds.

Live demo and supplementary video. The anonymous repository also contains a live demo of this workflow (`experiments/braque/demo_live`, served locally by `experiments/braque/scripts/09_serve_live_demo.py`; Figure 11). On each request, it runs a fresh ibUMAP or umap-learn fit followed by HDBSCAN on a fixed subset of the frozen L2 matrix; subsets of 1,000 to 20,000 cells are prefixes of one fixed permutation. The adjustable parameters default to the case-study values: `n_neighbors`, `min_dist`, `min_cluster_size` (with `min_samples` set equal), and deterministic mode with a seed. The number of epochs, spectral initialization, and the remaining HDBSCAN settings stay fixed. As in the case study, seeded umap-learn uses one job, whereas seeded ibUMAP runs in deterministic mode with the same number of jobs as its unseeded runs. After each run, the interface reports embedding and clustering times, the cluster count, the noise fraction, and a SHA-256 fingerprint of the coordinates and labels. From the second run on, it also reports the changed-assignment fraction relative to the previous completed run, computed with the matching used for Table 18.

The supplementary video summarizes the method and main results and ends with a screen recording of the demo on a 2,500-cell subset. The recording shows unseeded umap-learn reruns, seeded runs of both methods, and a change of `n_neighbors` followed by its reversal, after which the fingerprint equals that of the earlier seeded ibUMAP runs. The demo and the recording illustrate the protocol above. Their timings and change fractions come from single interactive runs on cell subsets and on a different machine, and are not among the reported results.



Figure 11: Live demo interface after a repeated seeded ibUMAP run on a 20,000-cell subset. Left: parameters and execution mode. Center: embedding colored by HDBSCAN cluster, with noise in gray. Right: comparison with the previous run (here 0.0% changed assignments and an identical fingerprint), runtimes, cluster count, noise fraction, and cluster sizes.