# Evaluating Accuracy and Performance Tradeoffs in GPU Accelerated Single Cell RNA-seq Analysis

Cory Gardner
Computer Science
Saint Louis University
St. Louis, MO, USA
cory.gardner@slu.edu

Seyun Jeong
Computer Science
Saint Louis University
St. Louis, MO, USA
seyun.jeong@slu.edu

Oam Khatavkar
Computer Science
Saint Louis University
St. Louis, MO, USA
oam.khatavkar@slu.edu

Aiden Moon
Parkway Central High School
Chesterfield, USA
yaimoon306@gmail.com

Qinglei Cao
Computer Science
Saint Louis University
St. Louis, MO, USA
qinglei.cao@slu.edu

Tae-Hyuk Ahn
Computer Science
Saint Louis University
St. Louis, MO, USA
taehyuk.ahn@slu.edu

## Abstract

Single-cell RNA sequencing (scRNA-seq) has rapidly become a big-data field, with experiments now profiling up to millions of cells in a single study. While GPU-accelerated pipelines (built on frameworks like NVIDIA RAPIDS and CuPy) promise to drastically reduce runtimes, questions remain about the fidelity and reproducibility of their outputs compared to traditional CPU-based workflows. We benchmarked matched CPU and GPU workflows on a public 1.3-million-cell dataset and systematically downsampled subsets. GPUs delivered a dramatic end-to-end speedup of over 10×, but this performance gain was accompanied by a measurable tradeoff in biological output. We found a consistent, moderate level of concordance between CPU and GPU clustering results (Adjusted Rand Index approximately 0.5) across all sample sizes. Furthermore, we demonstrated that clustering fidelity was relative to the platform used to generate the reference "ground truth." We also found that fidelity did not strongly correlate with sample size, suggesting that platform-specific algorithmic differences and parameter choices, such as a fixed feature space, were more significant drivers of variability than cell number alone. Together, these results indicated that while modern GPUs enabled fast, large-scale scRNA-seq analysis, researchers needed to be aware of the inherent tradeoffs and consider the choice of computational platform as a critical variable impacting the final biological interpretation.

## CCS Concepts

• **General and reference → Reliability**; • **Applied computing → Bioinformatics**; • **Computing methodologies → Parallel algorithms**.

## Keywords

Correctness, Reproducibility, High-Performance Computing, GPU, Bioinformatics, Single-Cell RNA-Seq, RAPIDS, Scanpy

**ACM Reference Format:**
Cory Gardner, Seyun Jeong, Oam Khatavkar, Aiden Moon, Qinglei Cao, and Tae-Hyuk Ahn. 2025. Evaluating Accuracy and Performance Tradeoffs in GPU Accelerated Single Cell RNA-seq Analysis. In *Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC Workshops '25), November 16–21, 2025, St Louis, MO, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3731599.3767378

## 1 Introduction

Recent advances in scRNA-seq have enabled profiling of gene expression at the resolution of individual cells, revolutionizing our understanding of cellular heterogeneity [10]. As technology improved, scRNA-seq dataset sizes have grown from thousands to over a million cells, creating a new scale of data that poses significant computational challenges. A landmark example is the 1.3 million mouse brain cell dataset from 10x Genomics [1]. Such large-scale single-cell studies have highlighted the need for high-performance computing (HPC) solutions in this domain, while simultaneously raising concerns about scalable, reproducible analysis [13, 26].

Widely-used scRNA-seq analysis frameworks like Seurat (in R) and Scanpy (in Python) provide comprehensive workflows for tasks such as normalization, dimensionality reduction, clustering, and visualization [20, 25]. Seurat and Scanpy have become standard tools for single-cell data analysis, but their implementations rely on CPU-based libraries (e.g., NumPy, SciPy, scikit-learn) and can be prohibitively slow or memory-intensive for datasets with hundreds of thousands or millions of cells [8, 17, 24]. As dataset sizes increase, the computational cost of these pipelines grows, often requiring hours to process millions of cells on high-end servers. This impedes interactive analysis and iterative exploration, motivating the exploration of faster, parallelized approaches.

To address performance bottlenecks, the community has turned to GPU acceleration for single-cell workflows. Modern GPUs offer thousands of cores and high memory bandwidth, which can accelerate matrix operations and graph algorithms common in single-cell analysis. Frameworks such as NVIDIA's RAPIDS suite and CuPy enable drop-in replacements for many CPU operations [16, 18]. Early demonstrations showed that scRNA-seq analyses can be sped up substantially using GPUs [4], and the scverse/RAPIDS ecosystem introduced *rapids-singlecell*, a GPU-accelerated single-cell analysis package designed to mirror Scanpy's API while offloading computations to CUDA GPUs [3, 4, 19]. More recent efforts like ScaleSC further improved scalability, enabling analysis of very large datasets on a single GPU by overcoming memory bottlenecks and removing cross-platform inconsistencies [9]. Crucially, ScaleSC emphasized the importance of *consistency* in results between GPU and CPU implementations and significantly improved scalability [9]. In parallel, GPU acceleration has also transformed genomic analysis workflows more broadly; for example, NVIDIA Parabricks demonstrates substantial speedups for DNA and RNA sequencing pipelines in production bioinformatics settings [15].

Despite these advances, fidelity and reproducibility remain central challenges. Numerical computations on different architectures (GPU vs. CPU) or even different parallel execution orders can yield subtle differences that potentially lead to divergent analysis outcomes. In scientific computing and HPC, it is well documented that floating-point non-associativity and non-deterministic parallel reductions can cause run-to-run variability and affect reproducibility [12]. In the context of single-cell analysis, we similarly observe that GPU-accelerated pipelines can produce outputs that do not exactly match those from a CPU pipeline given the same input data and algorithms. These discrepancies, while often numerically small, may impact downstream biological interpretations. Hu *et al.* (ScaleSC) reported that the output of Scanpy vs. rapids-singlecell pipelines differed in various steps, which they categorized as stemming from "system variance" (implementation differences between CPU and GPU algorithms) and "numerical variance" (floating-point rounding differences) [9]. Although neither type of variance implies a bug, even slight differences can lead to inconsistent clustering or gene marker identification, posing challenges for reproducibility.

We highlight three specific sources of pipeline discrepancy that emerged in our study (and in prior work [9]): *(i) PCA sign indeterminacy.* PCA is a key dimensionality reduction step where the sign of each principal component (eigenvector) is mathematically arbitrary. While standard CPU libraries often enforce a consistent sign convention, GPU implementations may differ, altering downstream steps that depend on PC directions [9, 17]. *(ii) Randomness in initialization and stochastic algorithms.* Many steps, from community detection (Leiden) to visualization using Uniform Manifold Approximation and Projection (UMAP), involve randomness. Parallel random number generators on GPUs can introduce non-determinism, as thread scheduling can yield different random sequences even with fixed seeds [9, 14, 22]. *(iii) Nearest-neighbor graph construction.* Clustering often proceeds by building a $k$-nearest neighbor ($k$NN) graph. Differences between approximate search methods common on CPUs and exact searches on GPUs can yield slightly different neighbor lists and thus different cluster boundaries [6, 9, 19].

To understand the systematic differences between CPU and GPU implementations, we examine the technical sources of platform variance in detail:

*PCA Sign Indeterminacy.* Standard CPU libraries (e.g., the PCA in scikit-learn used by Scanpy) enforce a consistent convention, such as flipping each eigenvector's sign so that the largest loading is positive [17]. In contrast, GPU-based PCA in RAPIDS cuML may not always perform this alignment, leading to sign-flipped PCs relative to Scanpy [9]. While mathematically equivalent for explained variance, these sign inconsistencies can systematically alter downstream steps that depend on PC directions, such as batch correction algorithms like Harmony [11].

*RNG and Thread Scheduling Effects.* Parallel random number generators on GPUs introduce nondeterminism through thread scheduling variations. Even with identical seeds, the order in which GPU threads execute can vary between runs, causing different sequences of random number generation [9]. This "RNG variance" propagates through stochastic algorithms like community detection (Leiden) and manifold learning (UMAP), potentially changing clustering boundaries and visualization layouts [14, 22].

*Neighbor Search Implementation Differences.* The choice between approximate (NN-descent on CPU) versus exact (brute-force GPU) neighbor search methods can yield systematically different $k$NN graphs [6, 19]. While both methods target the same mathematical result, approximate methods introduce controlled errors that can accumulate into measurably different clustering outcomes, especially at cluster boundaries where cells have ambiguous assignments.

*Floating-Point Non-Associativity.* Parallel reductions in matrix operations accumulate partial sums in data-dependent, thread-scheduled orders. Since $(a+b)+c \neq a+(b+c)$ in IEEE-754 arithmetic, different reduction tree structures yield slightly different results at the unit-in-the-last-place level [7, 12]. These minute differences can cascade through the pipeline, particularly affecting distance computations and optimization algorithms.

Motivated by these issues, our work provides a systematic, empirical comparison of matched CPU and GPU-accelerated scRNA-seq pipelines across a wide range of sample sizes from the 1.3 million cell dataset. We benchmark end-to-end performance and quantify the biological concordance of clustering outcomes using the Adjusted Rand Index similarity metric. Our findings reveal a significant performance-fidelity tradeoff: while the GPU provides a transformative speedup, the resulting cell clusters are measurably different from the CPU-based output. We further investigate the concept of fidelity by generating both CPU- and GPU-based references, demonstrating that the perceived accuracy of a platform is relative to the "ground truth" used for comparison. These results provide critical, practical guidance for researchers, highlighting the need to validate outputs when adopting accelerated workflows and underscoring that platform choice can be as consequential as sample size for ensuring reproducible biological conclusions.

## 2 Background: scRNA-seq and Analysis

scRNA-seq profiles gene expression at the resolution of individual cells, revealing cellular heterogeneity that bulk RNA-seq obscures [13, 26]. This capability has transformed biology, enabling discoveries in development, immunology, and cancer research. From a computational perspective, scRNA-seq produces large, sparse, high-dimensional count matrices, often with millions of cells and ~20,000 genes, posing significant storage and processing challenges [21, 27].

### Data Characteristics and Preprocessing

Modern droplet-based technologies such as the 10x Genomics Chromium system can capture tens of thousands of cells per experiment, and large consortia now routinely generate datasets exceeding one million cells [1]. The raw output is a gene-by-cell count matrix of unique molecular identifiers (UMIs), typically <10% dense. Standard preprocessing includes filtering low-quality cells and uninformative genes, per-cell normalization, log-transformation, and selection of highly variable genes (HVGs) for downstream analysis [20, 25].

### Canonical Analysis Workflow

The standard computational workflow for scRNA-seq follows a multi-step process [13, 20, 25, 26]:

(1) Preprocessing and Quality Control: Low-quality cells (e.g., with few detected genes or high mitochondrial gene fractions) and uninformative genes are filtered out. Counts are normalized per cell to correct for sequencing depth, and data are log-transformed for variance stabilization [21, 27].
(2) Feature Selection and Scaling: HVGs are selected to capture biological signal while reducing noise. Gene counts are then scaled to unit variance, often regressing out technical confounders such as sequencing depth [20, 25].
(3) Dimensionality Reduction (PCA): Principal component analysis (PCA) reduces the thousands of gene dimensions to 30–50 orthogonal components, which capture the dominant axes of variation across cells. PCA serves as the basis for downstream analyses.
(4) Neighborhood Graph Construction: Each cell is represented as a node in a *k*NN graph in PCA space. This graph encodes the local structure of the cellular manifold [6].
(5) Clustering: Graph-based community detection algorithms, such as Louvain or Leiden, partition cells into clusters corresponding to putative cell types or states [22].
(6) Nonlinear Embedding for Visualization: Algorithms such as UMAP or t-Distributed Stochastic Neighbor Embedding (t-SNE) map cells into two dimensions for visualization. These embeddings are not used for clustering directly but help interpret cell populations visually [2, 14, 23].
(7) Biological Interpretation: Differential expression analysis identifies marker genes for each cluster. These markers are cross-referenced with known cell-type signatures to assign biological meaning [20, 25].

### Computational Demands

From the perspective of high-performance computing, scRNA-seq workflows present several challenges:

- Scale: Datasets of millions of cells yield matrices with tens of billions of entries (even though sparse). Operations such as PCA, *k*NN search, and clustering must scale to these sizes [21].
- Heterogeneity of Workloads: The pipeline involves both dense linear algebra (e.g., PCA, regression) and irregular graph computations (*k*NN construction, community detection). These distinct workloads stress both CPU and GPU architectures differently [3, 9, 18].
- Stochasticity and Reproducibility: Many algorithms are nondeterministic (random initializations, parallel reductions), raising concerns about reproducibility across architectures and repeated runs [5, 7, 12].
- Memory Efficiency: Sparse matrices dominate scRNA-seq. Efficient storage formats and memory-aware algorithms are critical to avoid exceeding GPU memory capacity, especially at million-cell scale [4, 19].

## 3 Methods

### 3.1 Dataset and Preprocessing

All experiments used the publicly available *1.3 Million Mouse Brain Cells* dataset from 10x Genomics, profiling ~1.3M E18 mouse brain cells [1]. The raw UMI count matrix (genes × cells) was processed in Scanpy. Cells with extremely low gene counts and/or high mitochondrial transcript fractions were removed; genes detected in fewer than three cells were excluded. Counts were normalized to 10,000 per cell and log-transformed. For all analyses, the number of HVGs was fixed at 2,000. This was a deliberate methodological choice to control for feature space complexity, thereby isolating the specific impact of cell number and computational platform on clustering outcomes. Gene counts were scaled to unit variance, regressing out total counts and mitochondrial percentage. Principal Component Analysis (PCA) retained 50 components, which served as the basis for all downstream analyses.

### 3.2 Experimental Design and Subsampling

To systematically probe performance and stability, we performed a scaling analysis across a range of sample sizes. We generated 24 uniformly sampled subsets ranging from 50,000 to 1,200,000 cells in increments of 50,000, each with three independent replicates. For each subset size, we employed two sampling strategies: "random", where cells were chosen uniformly without replacement from the entire dataset, and "stratified", where cells were sampled to preserve the proportional cluster abundances observed in a high-resolution reference. Subsampling used fixed seeds to ensure identical cell lists across replicates and platforms for a given strategy.

To investigate the relativity of fidelity metrics, we generated two high-resolution biological references: one by running the full 1.2M cell subset through the GPU-accelerated pipeline, and a second by running the same subset through the CPU pipeline. The resulting Louvain clusters from these runs were designated as the "ground truth" for their respective fidelity analyses.

As a control experiment to assess the internal consistency of the GPU pipeline, we also performed a run-to-run stability test. For this, a fixed 25,000-cell subset was analyzed five times on the GPU, with each replicate using an identical random seed to measure numerical reproducibility under fixed conditions.

### 3.3 Pipeline Implementation and Determinism Controls

We implemented functionally equivalent CPU and GPU pipelines in Python 3.10. The CPU workflow used Scanpy (v1.9) with its standard backend of NumPy, SciPy, and scikit-learn; the GPU workflow used RAPIDS (v24.02), leveraging cuML for machine learning operations and cuGraph for graph analytics. Steps lacking direct GPU equivalents (e.g., specific regression models for differential expression) used CPU implementations in both workflows to ensure consistency.

To maximize comparability, all stochastic components, including PCA, clustering, and embedding initializations, were controlled with a fixed random seed that was propagated throughout each pipeline run, although fixed seeds were used, nondeterminism may still arise from GPU parallel reductions and floating-point accumulation order. PCA sign ambiguity was handled by applying a canonical sign convention to the eigenvectors. The primary workflow used for clustering was a $k$-nearest neighbors ($k = 15$) graph construction followed by Louvain community detection.

Figure 1 summarizes the end-to-end workflow used in this study, including matched CPU/GPU branches with identical parameters and seeds, the evidence packs we persist (PCA, $k$NN, embeddings, labels, timings), and the stability metrics (PCA fingerprint, $k$NN Jaccard, Procrustes RMSE, Adjusted Rand Index (ARI)).

## 3.4 Parameterization

Table 1 lists the key analysis parameters used across all experiments to ensure consistency between the CPU and GPU workflows.

**Table 1: Key analysis parameters.**

| Stage | Setting |
|---|---|
| HVGs | 2,000 (fixed for all runs) |
| Scaling/Regression | Center/scale; regress %mito and total counts |
| PCA | 50 components |
| Neighbors | $k = 15$; Euclidean distance in PCA space |
| Clustering | Louvain community detection |
| Differential Expression | Logistic regression (one-vs-rest) |
| Random Seed | Fixed globally for all stochastic steps |

## 3.5 Systems and Software

Analyses were run on a Linux-based high-performance computing system with a matched conda environment to ensure software consistency. Hardware and software summaries appear in Table 2.

**Table 2: Systems and software summary.**

| Component | Specification |
|---|---|
| CPU Nodes | 2x Intel Xeon 6548Y (64 cores), 512 GB Memory |
| GPU Nodes | NVIDIA H100 GPUs (80 GB), 1TB Memory |
| Operating System | Linux (x86_64) |
| Python | 3.10 |
| Scanpy | 1.9 |
| RAPIDS | 24.02 |

## 3.6 Performance and Stability Metrics

For each major step in the pipelines, we recorded wall-clock time and peak memory usage. CPU metrics were monitored with `psutil`, while GPU metrics were captured with `pynvml`. Unless stated otherwise, results are reported as the mean ± standard deviation over three replicates.

To compare clustering outcomes, we used the Adjusted Rand Index (ARI), a metric that measures the similarity between two data clusterings while correcting for chance. An ARI of 1.0 indicates perfect agreement, while an ARI of 0 indicates similarity is no better than random. We defined two key analyses using this metric:

- **Concordance:** The ARI calculated between the cluster labels produced by the CPU pipeline and the GPU pipeline for the same input cell subset. This measures the direct agreement between the two platforms.
- **Fidelity:** The ARI calculated between the cluster labels from a given subset run and the labels for those same cells as found in the corresponding 1.2M-cell "ground truth" reference (either CPU- or GPU-generated). This measures how well a smaller analysis recapitulates the full-scale result.

## 3.7 Best Practices for Reproducibility

Based on our experience, we compiled a checklist of best practices for ensuring reproducibility in accelerated scRNA-seq workflows (Table 3). These recommendations focus on controlling sources of variance and creating a thorough record of the computational experiment.

## 4 Results

## 4.1 GPU Acceleration Provides Major Performance Gains

GPU acceleration reduced runtime by more than an order of magnitude compared to the CPU workflow, with the performance gap widening as dataset size increased (Figure 2A). At 1.2 million cells, the CPU pipeline required nearly 3 hours, while the GPU pipeline completed in approximately 17 minutes. For computationally intensive steps such as neighborhood graph construction and clustering, the GPU speedup factor exceeded 150x over the multicore CPU implementation (Figure 2B), confirming that these common bottlenecks are particularly well-suited for GPU parallelization.

## 4.2 CPU and GPU Pipelines Show Moderate Biological Concordance

To quantify the agreement between the platforms, we computed the Adjusted Rand Index (ARI) between CPU and GPU clustering results for each matched run. We observed a moderate, yet consistent, level of concordance, with a median ARI between 0.45 and 0.55 across all sample sizes (Figure 3). This indicates that while the pipelines produce broadly similar clustering structures, there are significant, systematic differences in how they assign individual cells to clusters. The level of concordance did not show a clear trend with increasing sample size, suggesting the disagreement is an inherent property of the differing software implementations rather than an effect of data scale.

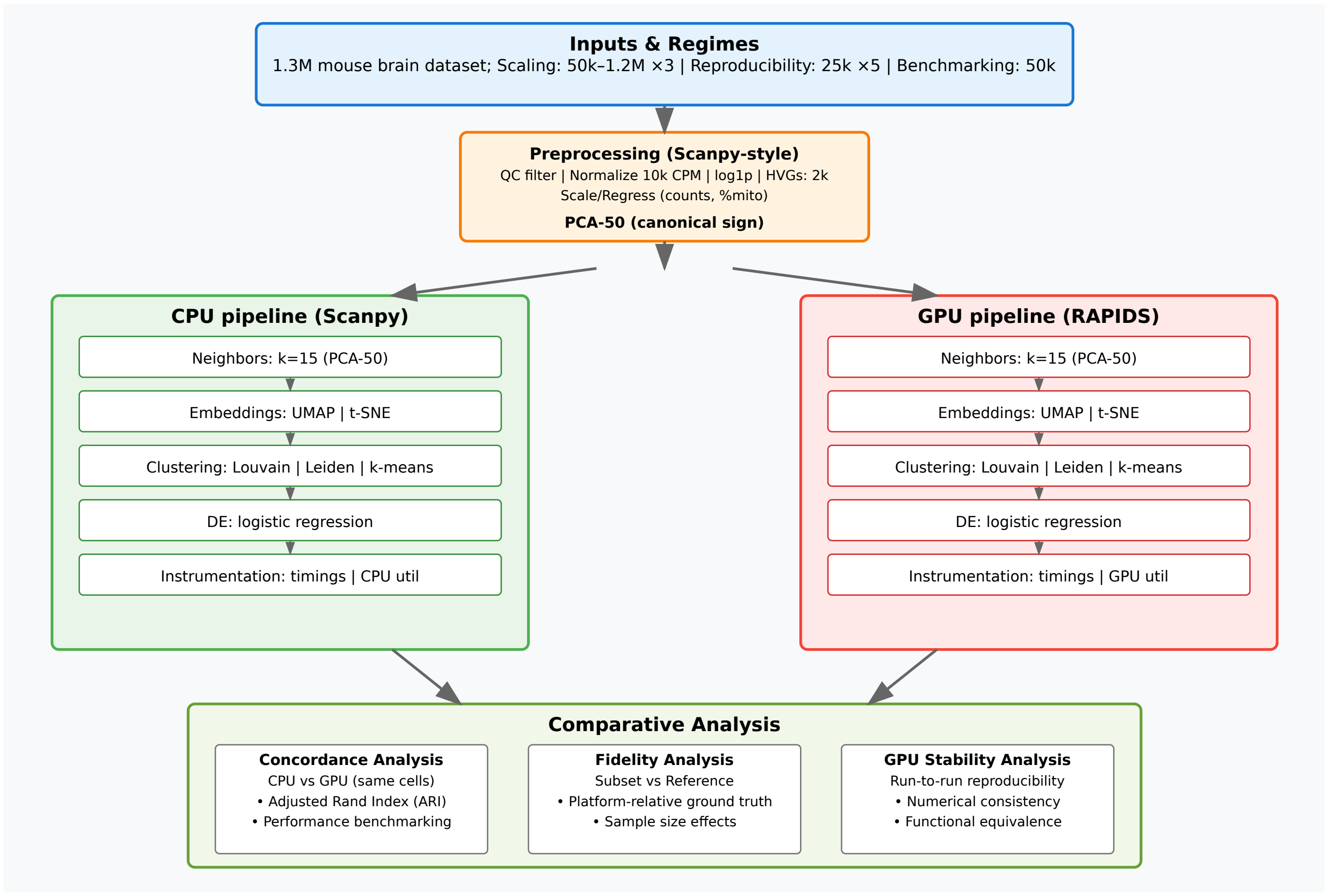

Inputs & Regimes
1.3M mouse brain dataset; Scaling: 50k–1.2M ×3 | Reproducibility: 25k ×5 | Benchmarking: 50k
Preprocessing (Scanpy-style)
QC filter | Normalize 10k CPM | log1p | HVGs: 2k
Scale/Regress (counts, %mito)
PCA-50 (canonical sign)
CPU pipeline (Scanpy)
Neighbors: k=15 (PCA-50)
Embeddings: UMAP | t-SNE
Clustering: Louvain | Leiden | k-means
DE: logistic regression
Instrumentation: timings | CPU util
GPU pipeline (RAPIDS)
Neighbors: k=15 (PCA-50)
Embeddings: UMAP | t-SNE
Clustering: Louvain | Leiden | k-means
DE: logistic regression
Instrumentation: timings | GPU util
Comparative Analysis
Concordance Analysis
CPU vs GPU (same cells)
• Adjusted Rand Index (ARI)
• Performance benchmarking
Fidelity Analysis
Subset vs Reference
• Platform-relative ground truth
• Sample size effects
GPU Stability Analysis
Run-to-run reproducibility
• Numerical consistency
• Functional equivalence

**Figure 1: End-to-end workflow for the comparative benchmark. After initial preprocessing, identical cell subsets were run through parallel CPU (Scanpy) and GPU (RAPIDS) pipelines. The primary outputs were performance logs and processed data files, which were analyzed using the ARI metric to measure concordance between platforms and fidelity against full-dataset references.**

**Table 3: Reproducibility checklist and recommended defaults for accelerated scRNA-seq.**

| Item | What to Control | Recommended Setting / Artifact |
|---|---|---|
| Environment Pinning | Exact versions for Python, Scanpy, RAPIDS, CUDA | Conda environment file; record package list and GPU driver |
| Randomness | Shared global seed propagated to all stochastic steps | Fixed SEED; pass `random_state` to all steps; record in metadata |
| PCA Sign Convention | PCs are sign-ambiguous by definition | Apply a canonical sign flip (deterministic rule); store final loadings |
| Neighbor Graph | Differences between exact/approximate search | Use one method consistently; record $k$, metric, and implementation |
| Clustering | Potential non-determinism in community detection | Fix algorithm seed and resolution; log the final labels |
| Hardware Metadata | Node/GPU/CPU details affect performance | Record GPU model/memory, CPU model, RAM, and driver versions |
| Evidence Pack | Inputs for later audit and validation | Save processed data, final labels, performance logs, and key metrics |

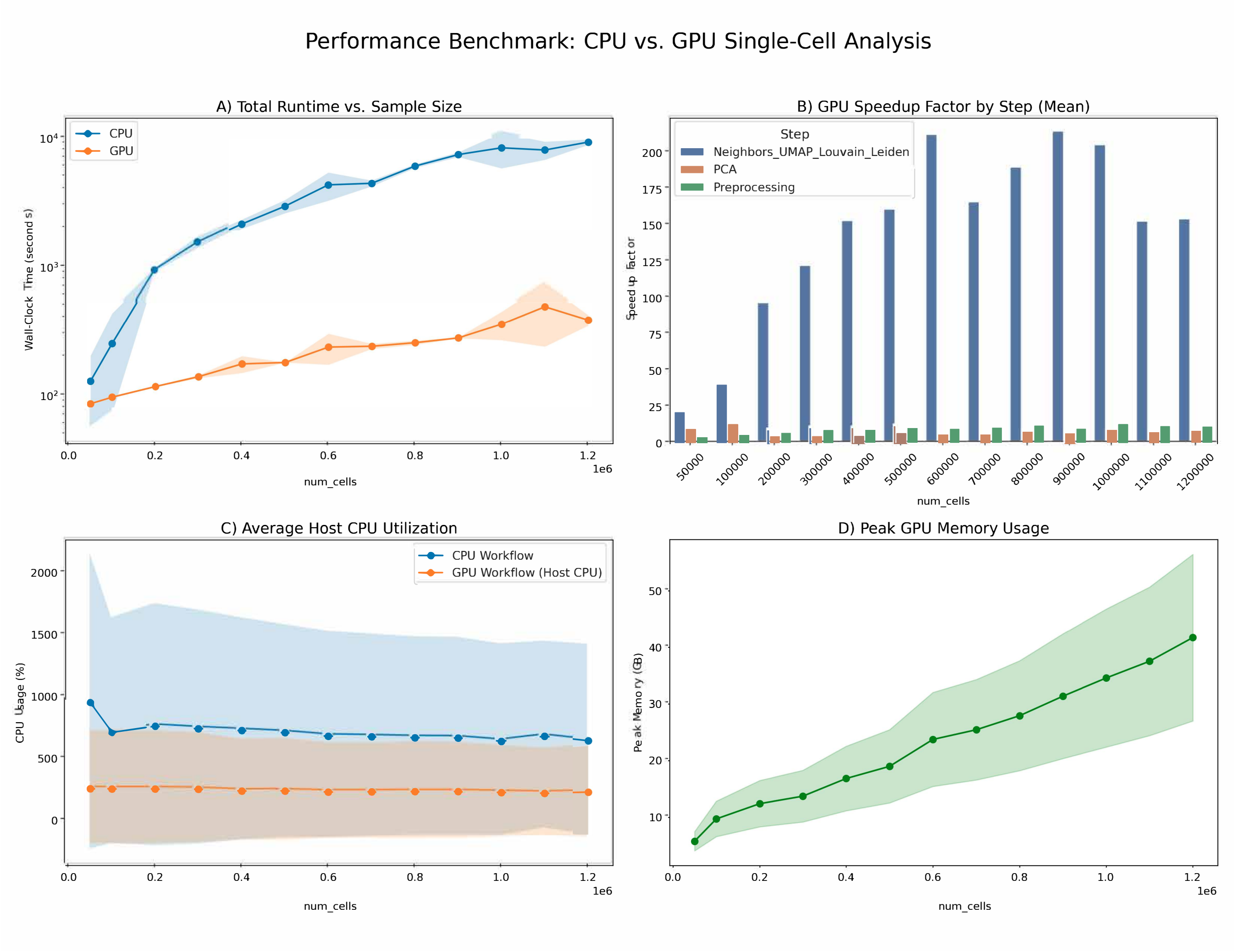


**Figure 2: Performance Benchmark: CPU vs. GPU. (A) Total wall-clock runtime vs. sample size. The GPU pipeline is consistently over 10x faster. (B) GPU speedup factor by step, showing the largest gains in graph-based algorithms.**

## 4.3 Clustering Fidelity is Relative to the Reference Platform

We next assessed how faithfully each platform's subset results recapitulated a "ground truth" clustering from the full 1.2M-cell dataset. Our analysis reveals that the concept of fidelity is entirely relative to the platform used to generate the reference. When measured against a GPU-generated reference (Figure 4A), the GPU pipeline consistently achieved higher fidelity scores. Conversely, when measured against a CPU-generated reference (Figure 4B), the CPU pipeline appeared superior. This demonstrates that neither platform is inherently more "correct," but rather that each is most consistent with itself, highlighting the challenge of defining an objective "ground truth" when comparing distinct computational ecosystems.

## 4.4 Fidelity Does Not Scale with Sample Size Due to Fixed Features

A key, and perhaps counterintuitive, finding from our study is that for both platforms, clustering fidelity did not meaningfully increase with sample size (Figure 4). The fidelity curves remain surprisingly flat across the full range of subsets from 50,000 to 1.2 million cells. We attribute this to our controlled experimental design, where the number of HVGs was fixed at 2,000 for all runs. This created an informational bottleneck, demonstrating that the biological resolution of the analysis was limited more by the fixed feature space than by the number of cells.

## 4.5 Visual Emergence of Local Cluster Structure

While the global fidelity of clustering did not improve with cell number, we observed that the visual cohesion of individual cell populations can still benefit from larger sample sizes. A case study

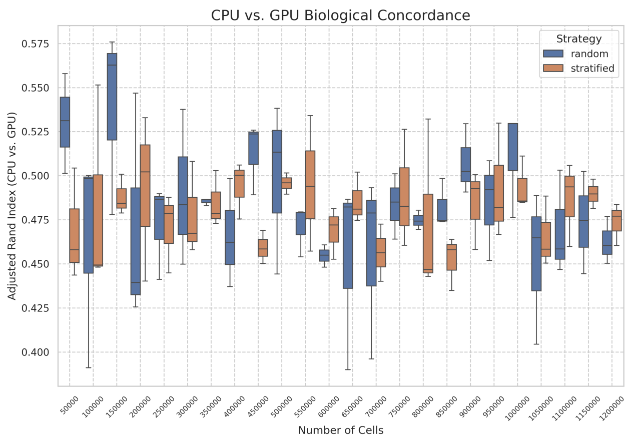


**Figure 3: CPU vs. GPU Biological Concordance. The ARI between CPU and GPU clusterings is consistently moderate (approximately 0.50) across all sample sizes and for both random and stratified sampling strategies.**

of a single reference cluster illustrates this effect (Figure 5). At 100k cells, the cells corresponding to this reference population were scattered and fragmented in the UMAP embedding. At 500k cells, the population began to coalesce into a more defined group, and at 1.2M cells, it formed a tight, distinct cluster. This demonstrates that while adding more cells can refine the local density and visual separation of known populations, this local improvement does not necessarily translate to an increase in the global accuracy of the entire partitioning when the feature space is constrained.

## 4.6 The GPU Pipeline is Numerically Stable

To establish that the moderate CPU-GPU concordance reflects systematic algorithmic differences rather than GPU-specific numerical instability, we conducted a controlled stability analysis of the GPU pipeline. Using a fixed 25,000-cell subset, we performed five independent runs with identical random seeds and parameter settings. The results demonstrate exceptional numerical reproducibility: PCA outputs showed near-perfect consistency with sign-invariant cosine similarities of 1.0000 (median) across all principal components, and per-cell embeddings differed by a median $L_2$ distance of only 0.0002 in 50-dimensional PCA space (Table 4). Downstream clustering outcomes were similarly stable, with a median Leiden ARI of 0.927 between replicate runs and perfect conservation of k-nearest neighbor graphs.

Figure 6 illustrates this stability: two independent GPU runs with identical seeds yield nearly indistinguishable t-SNE + k-means layouts after alignment. The preservation of global structure and cluster boundaries further supports the conclusion that GPU variability arises only from expected floating-point effects rather than instability. The minimal observed variations are consistent with expected unit-in-the-last-place differences from floating-point non-associativity in parallel reductions. This analysis confirms that the GPU-accelerated workflow is largely deterministic and internally consistent, establishing that the moderate CPU-GPU concordance (ARI $\approx$ 0.50) represents fundamental differences in algorithmic implementation and numerical libraries rather than stochastic noise or computational unreliability. These findings validate the GPU platform as a stable foundation for reproducible analysis while reinforcing that the choice between CPU and GPU constitutes a systematic analytical decision with measurable biological consequences.

**Table 4: PCA run-to-run fingerprint (25k subsample, 5 GPU runs, fixed seed).**

| Metric | Definition | Value |
|---|---|---|
| PC Cosine (median) | $\lvert\langle \mathbf{v}_i, \mathbf{w}_i \rangle\rvert$ | 1.0000 |
| PC Cosine (min) | Worst PC over all pairs | 1.0000 |
| Per-cell $L_2$ (median) | $\lVert \mathbf{z}_c^{(A)} - \mathbf{z}_c^{(B)} \rVert_2$ in 50D | 0.0002 |

## 5 Discussion

Our study evaluates GPU-accelerated scRNA-seq analysis from three complementary angles: (i) performance at scale, (ii) biological concordance between CPU and GPU workflows, and (iii) the numerical stability of the GPU pipeline. First, GPUs delivered substantial speedups, exceeding an order of magnitude end-to-end with stepwise gains >150x for graph construction (Figure 2). Second, this performance gain was accompanied by a measurable tradeoff in biological output: CPU–GPU concordance in clustering was moderate (mean ARI approximately 0.50), indicating that the accelerated workflow produces a systematically different result (Figure 3). Third, we found that the perceived fidelity of each platform is relative to the "ground truth" used for evaluation; each platform appeared most faithful when compared against a reference it generated itself (Figure 4). Together, these results support GPUs as a practical path to large-scale analysis, but underscore that platform choice is a critical variable impacting the final interpretation.

The widening GPU advantage with increasing cell counts mirrors the algorithmic mix of scRNA-seq: dense linear algebra (PCA), memory- and bandwidth-bound $k$NN graph construction, and iterative graph methods. These workloads map well to modern GPUs, which combine massive thread parallelism and high memory bandwidth. The observed linear scaling of peak GPU memory with sample size offers a concrete provisioning rule-of-thumb for analyses at the million-cell scale. Whole-pipeline timing, not just peak FLOPs, should guide hardware selection, as CPU-bound components and I/O can still represent a significant portion of the total runtime in certain regimes.

A central finding of our work is the existence of a significant performance-fidelity tradeoff. The moderate concordance (ARI of approximately 0.50) is not due to stochastic noise from the GPU; our run-to-run stability analysis confirms the GPU pipeline is highly deterministic, producing nearly identical clustering results on repeated runs with a fixed seed (Leiden ARI ~0.93). Therefore, the disagreement with the CPU is a result of systematic differences arising from distinct numerical libraries and algorithmic implementations. This finding does not imply one platform is more "correct" but demonstrates that they are not interchangeable.

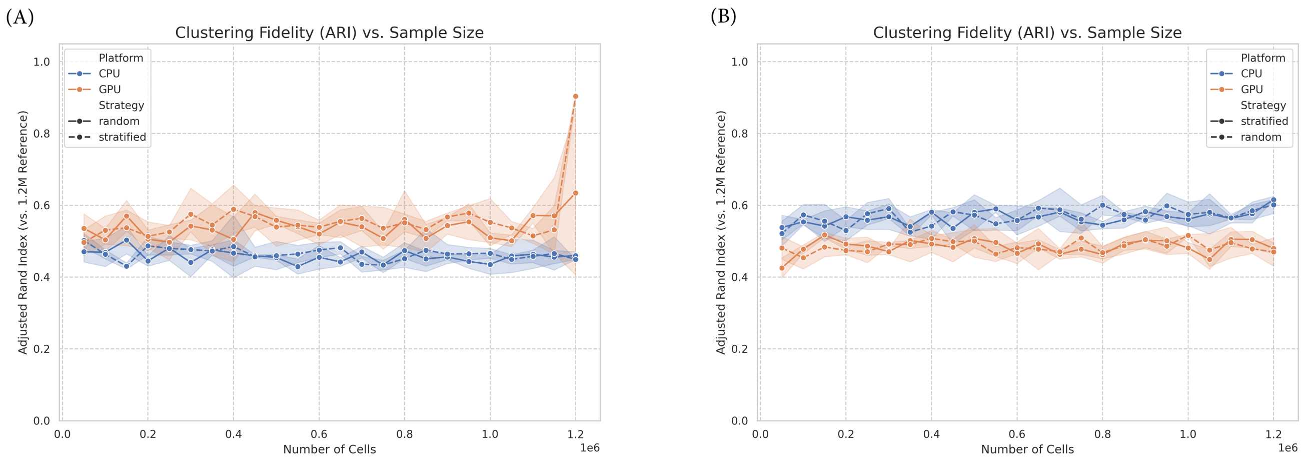


**Figure 4: Clustering Fidelity is Relative to the Reference Platform. (A) When measured against a GPU-generated reference, the GPU pipeline (orange) shows higher fidelity. (B) When measured against a CPU-generated reference, the CPU pipeline (blue) shows higher fidelity. In both cases, fidelity remains relatively flat across sample sizes.**

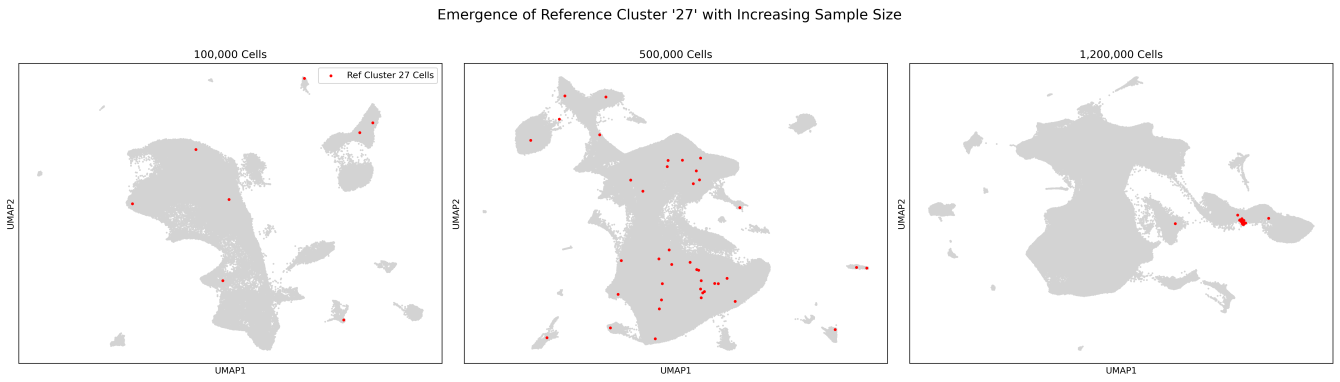
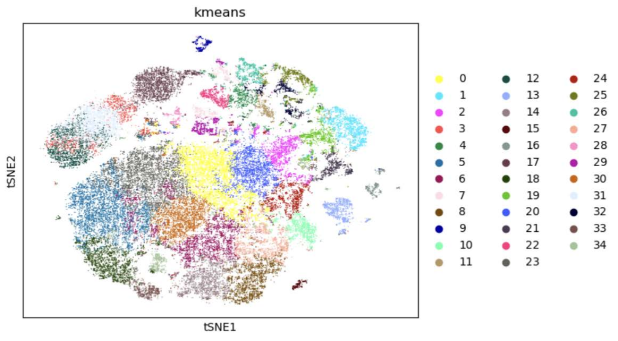


**Figure 5: Emergence of a stable cluster with increasing sample size. A single reference cell population (red) is visualized across UMAP embeddings of increasing cell number. (Left) At 100k cells, the population is fragmented. (Middle) At 500k cells, it begins to coalesce. (Right) At 1.2M cells, it forms a well-defined cluster.**

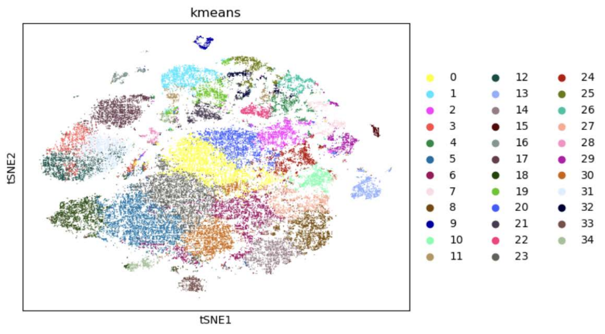
Beyond raw performance, our results challenge the simple assumption that increasing sample size uniformly improves cluster stability. The fidelity of subset analyses relative to the 1.2M-cell reference remained surprisingly flat across all sample sizes (Figure 4). We attribute this to our controlled experimental design, where the number of HVGs was fixed at 2,000. This created an informational bottleneck, demonstrating that the biological resolution was limited more by the feature space than by cell number. This illustrates a critical concept: true populations may fail to resolve not because of insufficient cell numbers, but because the features used for analysis do not contain enough information to separate them. Practically, analysts should consider that parameter tuning (like the number of HVGs) may be as important as sequencing depth for achieving stable results.

Based on our experience, we recommend:

(1) Acknowledge the tradeoff: Recognize that accelerated platforms may not produce bitwise identical results to CPU-based counterparts. Budget for validation and concordance analysis when adopting new tools.
(2) Report functional stability: Use metrics like ARI to quantify concordance. When assessing fidelity, be transparent about the origin of the "ground truth" reference, as it can bias the outcome.
(3) Match algorithms where possible: Differences in neighbor search or clustering implementations can dominate variance; align methods when making fidelity claims.
(4) Scale-aware workflow design: Use GPUs interactively for parameter tuning on moderate subsamples, then scale to the full dataset for final analysis.



**(a) Run A: t-SNE + $k$-means**



**(b) Run B: t-SNE + $k$-means**

**Figure 6: Two GPU runs with identical seed. Layouts are compared after alignment, and clusters are stable.**

(5) Holistic performance profiling: Optimize the whole pipeline, including I/O and CPU-bound steps, and choose hardware based on end-to-end time and memory requirements.

Our benchmarking used one dataset (E18 mouse brain) and a specific workflow (Scanpy-compatible with RAPIDS acceleration). While representative, performance and stability may vary with tissue complexity or alternate analysis methods. Our finding regarding the flat fidelity curve is tied to our deliberate choice of a fixed number of HVGs; future work should explore how fidelity scales when the feature space is allowed to grow with the sample size. Certain steps (such as differential expression) lacked GPU-native implementations and ran on CPU, potentially understating achievable end-to-end speedups as GPU library coverage improves.

Our results couple performance scaling with a robust analysis of concordance and the relativity of fidelity metrics. Immediate extensions include multi-GPU execution for datasets with $> 10^7$ cells and a deeper investigation into the biological drivers of discordance, for instance, by analyzing marker gene stability for matched clusters. More broadly, scRNA-seq remains a key benchmark domain for designing high-throughput, reproducible workflows in heterogeneous computing environments.

## 6 Future Work

Several extensions of this work would strengthen our understanding of CPU-GPU tradeoffs in single-cell analysis and provide more comprehensive guidance for practitioners.

*Dynamic Feature Space Analysis.* One limitation of our current study is the fixed number of highly variable genes (2,000), which created an artificial information bottleneck. Future work should systematically vary HVG counts with sample size to determine whether the flat fidelity curves persist under more realistic analytical conditions. This analysis would reveal whether larger datasets truly benefit from expanded feature spaces and how this impacts CPU-GPU concordance patterns.

*Cross-Platform Validation with Alternative References.* To further investigate the relativity of fidelity metrics, we propose incorporating additional computational frameworks as independent references. Analyzing the same datasets with Seurat (R-based) would provide a third reference point and help determine whether moderate concordance is specific to the Scanpy-RAPIDS comparison or represents a broader phenomenon across different implementation ecosystems. This triangulation approach could identify which biological conclusions are robust across platforms versus implementation-dependent.

*Comprehensive Memory Profiling.* While we characterized GPU memory scaling, systematic CPU memory profiling would complete the resource utilization picture. Peak memory consumption, memory bandwidth utilization, and scaling behavior across sample sizes would inform hardware provisioning decisions and identify memory-bound steps that could benefit from optimization on both platforms.

*Dataset Generalizability.* Our analysis focused on a single mouse brain dataset with known characteristics. Validation across diverse tissue types, species, and experimental protocols would establish the generalizability of our findings. Datasets with different sparsity patterns, cell type complexity, and batch effects could reveal whether CPU-GPU concordance varies with biological context, potentially identifying scenarios where platform choice is more or less consequential.

*Mechanistic Analysis of Discordance Sources.* A deeper investigation into the specific algorithmic and numerical sources driving the observed differences would provide actionable insights for improving cross-platform consistency. Systematic perturbation experiments isolating individual pipeline components (PCA implementation, neighbor search methods, clustering algorithms) could quantify the relative contribution of each factor to overall discordance and guide targeted improvements.

*Biological Impact Assessment.* Moving beyond clustering metrics to assess how CPU-GPU differences propagate to downstream biological conclusions would provide crucial practical guidance. Comparative analysis of differential gene expression, pathway enrichment, and cell trajectory inference could identify which biological interpretations are robust to platform choice versus sensitive to implementation details.

*Large-Scale Multi-GPU Analysis.* Extending analysis to datasets exceeding 10 million cells using distributed GPU computing would test whether our conclusions hold at emerging data scales and evaluate the scalability limits of current GPU-accelerated frameworks.

*Algorithm-Agnostic Comparisons.* Testing alternative clustering algorithms (hierarchical methods, density-based approaches) and dimensionality reduction techniques beyond PCA would establish whether moderate concordance is specific to graph-based workflows or represents a fundamental characteristic of CPU-GPU numerical differences. These investigations would collectively provide a more complete understanding of when and why computational platform choice matters for single-cell analysis, ultimately enabling evidence-based hardware and software selection for reproducible biological discovery.

## 7 Conclusion

Modern GPUs deliver substantial, scalable acceleration for end-to-end scRNA-seq analysis, representing a critical step forward for handling massive datasets. Our work demonstrates, however, that this performance comes with a measurable tradeoff in biological output. We found that CPU and GPU pipelines produce clustering results with only moderate agreement and that the perceived fidelity of each platform is relative to the chosen computational "ground truth." Furthermore, we highlight that analysis parameters, such as the number of features selected, can be a more dominant factor in reproducibility than sample size alone. These findings underscore the need for practitioners to treat the computational platform not as a simple implementation detail, but as a key variable that requires careful consideration and validation to ensure robust and reproducible scientific conclusions.

## References

[1] 10x Genomics. 2017. 1.3 Million Brain Cells from E18 Mice. https://www.10xgenomics.com/datasets/1-3-million-brain-cells-from-e-18-mice-2-standard-1-3-0.

[2] Etienne Becht, Leland McInnes, John Healy, Charles-Antoine Dutertre, Iain WH Kwok, Lai Guan Ng, Florent Ginhoux, and Evan Wing Newell. 2019. Dimensionality reduction for visualizing single-cell data using UMAP. *Nature Biotechnology* 37 (2019), 38–44. doi:10.1038/nbt.4314

[3] Scott Black, David Glicksberg, Corey Johnson, Andrew Meadows, Tom Dunn, Folly Combes, Montse Torras, Michael D'Amato, M. Jordan Rowley, Chittibabu Guda, Paul L. Sorgen, Laura Beaudry, Dale Hawkins, Allen Kim, Cory Gardner, and Tae-Hyuk Ahn. 2022. GPU-accelerated single-cell analysis with RAPIDS. *bioRxiv* (2022). doi:10.1101/2022.05.26.493607

[4] NVIDIA Developer Blog. 2023. GPU-accelerated single-cell RNA analysis with RAPIDS singlecell. https://developer.nvidia.com/blog/gpu-accelerated-single-cell-rna-analysis-with-rapids-singlecell/. Accessed 2025-08-23.

[5] Brett Daley. 2019. Comment on "GPU is non-deterministic?". https://github.com/openai/baselines/issues/805. Accessed: August 3, 2025.

[6] Wei Dong, Charikar Moses, and Kai Li. 2011. Efficient K-nearest neighbor graph construction for generic similarity measures. In *Proceedings of the 20th International Conference on World Wide Web Companion*. 577–586. doi:10.1145/1963405.1963487

[7] David Goldberg. 1991. What Every Computer Scientist Should Know About Floating-Point Arithmetic. *Comput. Surveys* 23, 1 (1991), 5–48. doi:10.1145/103162.103163

[8] Charles R Harris, K Jarrod Millman, Stéfan J van der Walt, et al. 2020. Array programming with NumPy. *Nature* 585 (2020), 357–362. doi:10.1038/s41586-020-2649-2

[9] Yiqiao Hu, Jinghan Zhang, Zhuowei Chen, et al. 2025. ScaleSC enables scalable and consistent single-cell analysis on GPUs. *Bioinformatics Advances* 5, 1 (2025), vbaf167. doi:10.1093/bioadv/vbaf167

[10] D. Jovic, X. Liang, H. Zeng, L. Lin, F. Xu, and Y. Luo. 2022. Single-cell RNA sequencing technologies and applications: A brief overview. *Clinical and Translational Medicine* 12, 3 (March 2022), e694. doi:10.1002/ctm2.694

[11] Ilya Korsunsky, Nicholas Millard, Jean Fan, Kamil Slowikowski, Fan Zhang, Kevin Wei, Yury Baglaenko, Michael Brenner, Po-Ru Loh, and Soumya Raychaudhuri. 2019. Fast, sensitive and accurate integration of single-cell data with Harmony. *Nature Methods* 16 (2019), 1289–1296. doi:10.1038/s41592-019-0619-0

[12] Ignacio Laguna. 2020. Varity: Quantifying Floating-Point Variations in HPC Systems Through Randomized Testing. In *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 622–633. doi:10.1109/IPDPS47924.2020.00070

[13] Malte D Luecken and Fabian J Theis. 2019. Current best practices in single-cell RNA-seq analysis: a tutorial. *Molecular Systems Biology* 15, 6 (2019), e8746. doi:10.15252/msb.20188746

[14] Leland McInnes, John Healy, and James Melville. 2018. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *arXiv preprint arXiv:1802.03426* (2018).

[15] Kevin A. O'Connell, Zaryab B. Yosufzai, Robert A. Campbell, et al. 2023. Accelerating genomic workflows using NVIDIA Parabricks. *BMC Bioinformatics* 24, 1 (2023), 221. doi:10.1186/s12859-023-05292-2

[16] Ryosuke Okuta, Yuya Unno, Daisuke Nishino, Satoshi Hido, and Crissman Loomis. 2017. CuPy: A NumPy-Compatible Library for NVIDIA GPU Calculations. In *Proceedings of Workshop on Machine Learning Systems (LearningSys) in NIPS 2017*. https://cupy.dev/

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, et al. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[18] RAPIDS AI. 2024. RAPIDS: Open GPU Data Science. https://rapids.ai. Accessed 2025-08-23.

[19] scverse and RAPIDS. 2024. rapids-singlecell. https://github.com/rapidsai/rapids-singlecell. Accessed 2025-08-23.

[20] Tim Stuart, Andrew Butler, Paul Hoffman, Christoph Hafemeister, Efthymia Papalexi, William M Mauck III, Yuhan Hao, Marlon Stoeckius, Peter Smibert, and Rahul Satija. 2019. Comprehensive integration of single-cell data. *Cell* 177, 7 (2019), 1888–1902.e21. doi:10.1016/j.cell.2019.05.031

[21] Valentine Svensson, Roser Vento-Tormo, and Sarah A. Teichmann. 2018. Exponential scaling of single-cell RNA-seq in the past decade. *Nature Protocols* 13, 4 (2018), 599–604. doi:10.1038/nprot.2017.149

[22] Vincent A Traag, Ludo Waltman, and Nees Jan van Eck. 2019. From Louvain to Leiden: guaranteeing well-connected communities. *Scientific Reports* 9, 1 (2019), 5233. doi:10.1038/s41598-019-41695-z

[23] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. In *Proceedings of the 20th International Conference on Machine Learning*. 243–250.

[24] Pauli Virtanen, Ralf Gommers, Travis E Oliphant, et al. 2020. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods* 17 (2020), 261–272. doi:10.1038/s41592-019-0686-2

[25] F. Alexander Wolf, Philipp Angerer, and Fabian J. Theis. 2018. Scanpy: large-scale single-cell gene expression data analysis. *Genome Biology* 19, 1 (2018), 15. doi:10.1186/s13059-017-1382-0

[26] Luke Zappia, Belinda Phipson, and Alicia Oshlack. 2018. Exploring the single-cell RNA-seq analysis landscape with the scRNA-tools database. *PLOS Computational Biology* 14, 6 (2018), e1006245. doi:10.1371/journal.pcbi.1006245

[27] Grace X. Y. Zheng, Jessica M. Terry, Phillip Belgrader, Paul Ryvkin, Zachary W. Bent, Rich Wilson, Solongo B. Ziraldo, T. D. Wheeler, Geoff P. McDermott, Junjie Zhu, M. T. Gregory, J. Shuga, L. Montesclaros, J. G. Underwood, D. A. Masquelier, S. Y. Nishimura, M. Schnall-Levin, P. W. Wyatt, C. M. Hindson, R. Bharadwaj, A. Wong, K. D. Ness, L. W. Beppu, H. J. Deeg, C. McFarland, K. R. Loeb, W. J. Valente, N. G. Ericson, E. A. Stevens, J. P. Radich, T. S. Mikkelsen, B. J. Hindson, and J. H. Bielas. 2017. Massively parallel digital transcriptional profiling of single cells. *Nature Communications* 8 (2017), 14049. doi:10.1038/ncomms14049