scProto: Affinity-Guided Prototype Learning for Cross-Batch Metacell Construction

Anonymous authors
Paper under double-blind review

Abstract

Resolving which cell states are present in a tissue is essential for characterizing cellular heterogeneity in development and disease. Single-cell and spatial transcriptomics profile many donors and sections at once, but individual cells are too sparse to analyze alone, so cells are aggregated into metacells. Current methods form metacells from expression proximity alone: a state rare in every batch but common overall cannot be formed, and a cell's spatial niche is either absorbed into cell-type signal or, if added to the affinity directly, overcorrects and splits one state across neighborhoods. We introduce scProto, a prototype-based deep clustering model that learns metacells across an entire experiment rather than within each batch, using the affinity graph as a training objective rather than clustering it directly: a conditional variational autoencoder learns a batch-corrected representation jointly with soft prototype assignment under a cohesion criterion that keeps each group internally dense via both direct and shared-neighborhood similarity. Assignments come from expression alone, so a niche separates cells only to the extent it changes their transcriptome. Across single-cell and spatial datasets, scProto recovers rare and niche-associated states missed by within-batch clustering, improving batch mixing and rare-population recovery while preserving neighborhood structure.

1 Introduction

Single-cell transcriptomics provides a high-resolution view of cellular heterogeneity, enabling the study of diverse cell states within complex tissues. However, these measurements are inherently sparse and noisy (Luecken & Theis, 2019), making it difficult to reliably recover the underlying biological states. A common approach to address this challenge is to aggregate similar cells into *metacells* (Baran et al., 2019), which provide denoised gene expression profiles that better reflect true cellular states. This relies on the assumption that variation within a group of similar cells is largely driven by noise, while shared patterns correspond to meaningful biological structure. Cell–cell affinity—a measure of pairwise similarity that guides which cells are grouped together—is central to this process and can in principle be derived from various biological sources.

Current metacell methods define this affinity based solely on gene expression and group cells that are close in this space. In multi-batch settings, batch effects distort gene expression, making similarity relationships reliable mainly within individual datasets. As a result, existing metacell methods (Baran et al., 2019; Persad et al., 2023; Ben-Kiki et al., 2022) form aggregations within individual batches and fail to generalize across them, even when underlying cell states are shared. This is particularly limiting for rare cell populations, where a single batch may not contain enough samples to recover stable cell states. Beyond transcriptomics, spatial organization shapes cellular programs through local interactions (Browaeys et al., 2020; Jin et al., 2021; Liu et al., 2026), yet existing methods ignore this signal entirely.

We introduce **scProto**, a framework that learns metacells by preserving affinity-defined relationships while remaining faithful to gene expression data (Figure 1). Given one or more affinity graphs, scProto encodes cells into a latent space and learns a set of shared prototypes representing metacells. Cells are probabilistically assigned to prototypes, and a cohesion objective rewards each cell for being similar to the other cells currently assigned to its own prototype, under both the affinity graph directly and a shared-neighborhood similarity derived from it. This drives highly connected cells to share a prototype, preserving the graph's local and neighborhood-level structure. Because the

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.

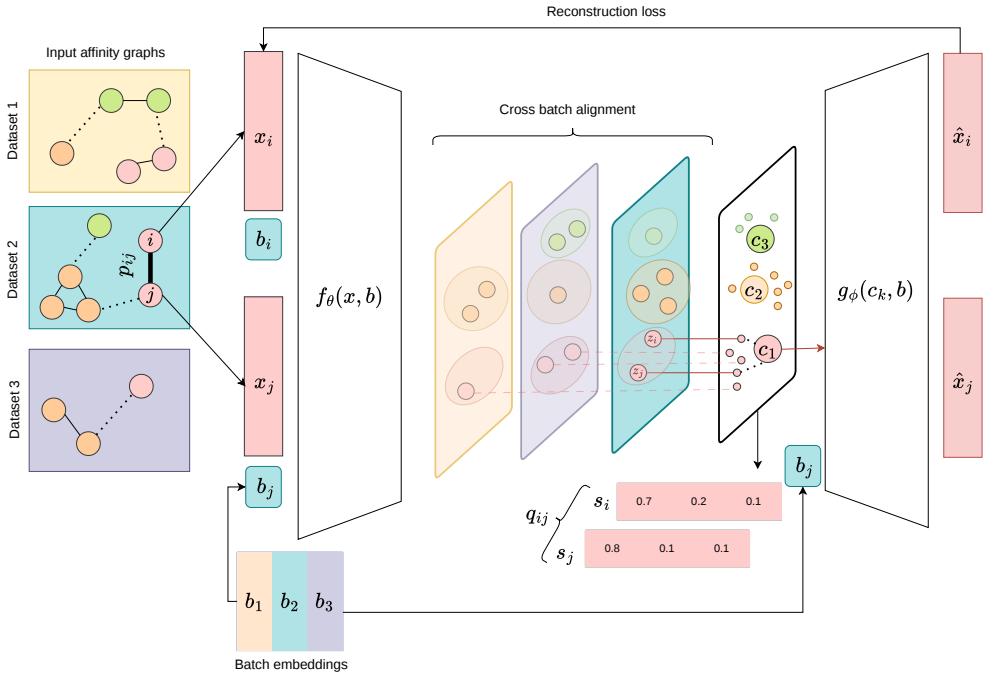


Figure 1: Overview of scProto. Cells from multiple batches are encoded into a shared latent space and softly assigned to learnable prototypes (metacells). A cohesion objective keeps each prototype's assignments dense under the input affinity graph; cross-batch alignment itself comes from a batch-conditioned reconstruction pretraining stage, which the model continues to preserve rather than re-derive once prototypes are introduced.

affinity is a plug-in component, the same model can preserve any biologically meaningful pairwise signal—whether transcriptomic neighborhood structure, spatial microenvironmental context, or any other relationship between cells—without architectural changes. At the same time, a reconstruction objective ensures that the learned representations capture meaningful gene expression patterns.

We evaluate scProto on four datasets spanning single-cell and spatial transcriptomics. Our results show that scProto forms metacells that integrate cells across batches while preserving within-batch structure, achieving high batch mixing without sacrificing modularity and cell-type purity. This leads to improved representation of rare cell populations, where cross-batch aggregation increases effective sample size and yields more stable metacells. The flexibility of the affinity framework extends naturally to spatial settings: by defining cell–cell affinity from spatial information, scProto captures microenvironmental structure more effectively than transcriptomics-only approaches, identifying metacells that reflect both cell identity and niche-specific variation.

Our contributions are as follows:

- We introduce a prototype-based framework that learns cross-batch metacells by jointly preserving affinity-defined similarity structure and maintaining fidelity to gene expression, with a cohesion-based training objective that keeps prototypes compact and prevents them from going empty.

- We demonstrate that affinity guidance, layered on top of a batch-corrected pretrained encoder, enhances representation of rare cell populations and enables effective integration of spatial context, without sacrificing the cross-batch alignment that pretraining already provides.

2 Related Work

Metacell methods. Metacell approaches address the sparsity and noise in single-cell data by aggregating highly similar cells into stable representative profiles. The standard pipeline constructs a cell–cell affinity graph—typically a k-nearest-neighbor graph with kernel-based similarity on PCA embeddings—and then partitions it into coherent groups. MetaCell (Baran et al., 2019) introduced this paradigm by partitioning a kNN graph into balanced, densely connected groups using community detection algorithms (Blondel et al., 2008; Traag et al., 2019) that operate on graph topology alone, without incorporating gene expression features directly. SEACells (Persad et al., 2023) refines the affinity with an adaptive Gaussian kernel that accounts for local density variations, and applies archetypal analysis (Cutler & Breiman, 1994) to identify extreme points on the data manifold as representative cell groups. Other methods such as Metacell-2 (Ben-Kiki et al., 2022) and SuperCell (Bilous et al., 2022) similarly aggregate cells based on transcriptomic graph structure, but like MetaCell, they remain restricted to within-batch similarity.

These methods share two limitations. First, because batch effects induce systematic shifts across datasets, the affinity graph is reliable only within individual batches, and metacells remain batch-specific even when the underlying cell states are shared across datasets. Second, groupings are defined purely by affinity structure, with no learning objective that validates whether the grouped cells share coherent gene expression patterns. MetaQ (Li et al., 2025) attempts to address the second limitation by replacing graph construction with end-to-end representation learning via a vector-quantization objective (van den Oord et al., 2017), but without an explicit batch correction mechanism the model can reduce its loss by learning batch-specific metacells—an always-available signal—rather than shared biological states. SURE (Cai et al., 2024) takes a generative approach that jointly constructs metacells and removes batch effects, but like MetaQ, optimizes reconstruction without affinity guidance, failing to preserve local cell–cell similarity structure—a limitation shared by reconstruction-based batch effect correction methods.

Batch effect correction. Dedicated reconstruction-based methods such as scVI (Lopez et al., 2018) use conditional variational autoencoders (Sohn et al., 2015) to disentangle biological variation from batch effects, but optimize a global reconstruction loss without affinity guidance, causing rare cell states to be absorbed into larger clusters. scPoli (De Donno et al., 2023) partially addresses this by anchoring prototypes to known cell types, though this requires accurate annotations, which are often incomplete or inconsistent in practice (Zeng, 2022). Non-VAE approaches such as Harmony (Korsunsky et al., 2019) and BBKNN (Polański et al., 2020) take different strategies—iterative embedding alignment and graph-level connectivity enforcement, respectively—but introduce their own distortions: Harmony can disrupt within-batch cell-type structure, while BBKNN assumes all cell types are present in every batch. Taken together, none of these methods incorporate an affinity graph during training—they align cells but do not explicitly preserve the community structure defined by cell–cell relationships within each dataset. Preserving such structure requires objectives that operate at the level of groups rather than individual cell distances, which connects to the literature on deep clustering.

Deep clustering and prototype-based learning. Deep clustering methods (Caron et al., 2018) jointly learn embeddings and cluster assignments, alternating between assigning cells to clusters and updating the network to predict those assignments. SeLa (Asano et al., 2019) replaces k-means with Sinkhorn–Knopp (Cuturi, 2013) assignment for balanced pseudo-labels; SwAV (Caron et al., 2020) extends this online with learnable prototypes. However, this enforces uniform usage across prototypes—which does not reflect the highly variable cluster sizes in biological data—and defines positive structure through data augmentation rather than input-defined similarity, providing no information about how similar two cells actually are; in single-cell settings, a cell–cell affinity graph provides a more principled source of structure. Graph-based methods such as Parametric UMAP (Sainburg et al., 2021) do incorporate such a graph, learning embeddings that preserve pairwise affinities, but target distances between individual cells rather than cluster-level assignments—in metacell learning, what matters is which cells are grouped together, not the precise geometry of the embedding space. This idea has also been applied to single-cell data, but still suffers from uniform prototype usage (Hashemi Golpayegani et al., 2025). A related failure mode across prototype-based methods is prototype collapse, where a subset of prototypes captures most cells and the rest go unused. Arteaga et al. (Arteaga et al., 2026) address this via decoupled prototype updates modelled as

a Gaussian mixture, though without enforcing distinctness, prototypes may remain redundant rather than capturing distinct clusters.

In summary, scProto addresses both limitations by combining affinity guidance with cross-batch alignment, without enforcing uniform prototype usage.

3 METHOD

We consider a dataset of cells $\{x_i\}_{i=1}^N$ with $B \geq 1$ batches, where $x_i \in \mathbb{R}^G$ denotes gene expression and $b_i \in \{1, \ldots, B\}$ is the batch label. The model consists of a batch-conditional encoder mapping cells to a shared latent space, a set of learnable prototypes representing metacells, and a batch-conditional decoder that reconstructs each cell directly from its own encoding, so that batch-specific variation is handled at decoding rather than absorbed into shared prototypes.

3.1 MODEL COMPONENTS

Input. We assume gene expression profiles $x_i \in \mathbb{R}^G$ with batch labels b_i, and a precomputed, sparse cell–cell similarity matrix (graph) $P \in \mathbb{R}^{N \times N}$, approximately block-diagonal by batch (construction and reasoning in Appendix B). By default P is an adaptive Gaussian (RBF) kernel on PCA embeddings (Persad et al., 2023), correcting for local density so rare cell types stay well-connected; disentangling the biological and batch-driven variation this mixes together is delegated to the batch-conditional encoder during training. More broadly, P can encode any biologically meaningful similarity—including spatial context—while the reconstruction objective keeps the model grounded in gene expression (Appendix C).

Encoder. A conditional encoder maps each cell to latent space:

$$z_i = f_\theta(x_i, b_i) \in \mathbb{R}^d. \tag{1}$$

Prototypes. We introduce K learnable prototypes $\{c_k\}_{k=1}^K$, $c_k \in \mathbb{R}^d$, representing metacells in the latent space.

Soft Assignment. Each cell is represented as a soft mixture of prototypes:

$$s_{ik} = \frac{\exp\left(\frac{z_i^\top c_k}{\tau}\right)}{\sum_{k'} \exp\left(\frac{z_i^\top c_{k'}}{\tau}\right)}, \tag{2}$$

where $\tau > 0$ is a temperature parameter controlling assignment sharpness, and both z_i and c_k are ℓ_2-normalized (see Appendix D). Rather than treating τ as a fixed hyperparameter, we calibrate it from data after pretraining and prototype initialization (see Appendix F).

The assignment vector $s_i \in \mathbb{R}^K$ encodes the soft membership of cell i across all K prototypes. This representation is used to preserve similarity structure by mapping highly connected cells to similar assignment vectors.

Community Structure Preservation. The idea is simple: cells the affinity graph calls related should end up with similar prototype assignments, and cells it calls unrelated should end up with dissimilar ones. We measure assignment similarity on a stop-gradient copy of the prototypes,

$$\hat{s}_{ik} = \frac{\exp\left(z_i^\top \mathrm{sg}(c_k)/\tau\right)}{\sum_{k'} \exp\left(z_i^\top \mathrm{sg}(c_{k'})/\tau\right)}, \qquad q_{ij} = \hat{s}_i^\top \hat{s}_j \in [0, 1], \tag{3}$$

where $\mathrm{sg}(\cdot)$ denotes a stop-gradient, so this loss can only move the encoder and never the prototypes directly (letting it move $\{c_k\}$ instead collapses most of them — Appendix D). We match q_{ij} to the graph with binary cross-entropy:

$$\mathcal{L}_{\text{community}} = -\sum_{i,j} \Big[P_{ij} \log \tilde{q}_{ij} + (1 - P_{ij}) \log(1 - \tilde{q}_{ij}) \Big], \tag{4}$$

with $\tilde{q}_{ij}$ clipped away from 0 and 1. We sample positive pairs from P_{ij}'s real edges, and negative pairs uniformly within the same batch only. Cross-batch alignment is handled instead by $\mathcal{L}_{\text{cVAE}}$ (Total Objective below).

Cohesion. We want similar cells to end up sharing a prototype. Concretely, we update the encoder and the prototypes jointly so that a cell's mean similarity to the other members of its own prototype is as high as possible. Since neither $\mathcal{L}_{\text{community}}$ nor $\mathcal{L}_{\text{cVAE}}$ touches the prototypes (previous paragraph; Total Objective below), this is the only term that shapes $\{c_k\}$ directly. We take the objective itself from the classical clustering literature rather than inventing a new one: it is the standard notion of cluster cohesion (Tan et al., 2005), equivalent to weighted kernel k-means / normalized association (Dhillon et al., 2004) once membership is soft. It uses the live assignment s_i (not the frozen $\hat{s}_i$ above), so its gradient reaches both the encoder and the prototypes.

Let $\mathrm{sim}(i, j)$ denote a similarity between cells i and j, with $\mathrm{sim}(i, i) := 1$ by convention. We use two such similarities in practice: the affinity graph directly, and a Gram-style similarity built from affinity rows that captures shared neighborhoods rather than direct connections (Appendix E defines both precisely). For prototype k, define its size and the average similarity a cell i has to k's current members:

$$\mathrm{size}_k = \sum_i s_{ik}, \qquad \overline{\mathrm{sim}}(i, k) = \frac{\sum_j s_{jk}\,\mathrm{sim}(i, j)}{\mathrm{size}_k}. \tag{5}$$

Cohesion is the assignment-weighted average of this quantity over all cells:

$$\mathrm{cohesion} = \frac{1}{N}\sum_i \sum_k s_{ik}\,\overline{\mathrm{sim}}(i, k), \qquad \mathcal{L}_{\text{cohesion}} = 1 - \mathrm{cohesion}, \tag{6}$$

so minimizing $\mathcal{L}_{\text{cohesion}}$ is exactly maximizing that mean similarity. Fixing $\mathrm{sim}(i, i) := 1$ is what avoids dead prototypes and lets the model use its full capacity. An empty prototype scores exactly 0; without a self term, a prototype that has only just recruited its first member would score 0 too — indistinguishable from empty, and with no reason for the model to ever grow it further. With $\mathrm{sim}(i, i) := 1$, that same prototype scores 1 instead, giving it a foothold to keep recruiting rather than being abandoned outright.

Estimating cohesion from a mini-batch. Scoring even one mini-batch cell against $\overline{\mathrm{sim}}(i, k)$ requires size_k and every other cell's current assignment — the full column of s, not just the mini-batch's. We keep this in a memory: an $N \times K$ table S^{mem} holding every cell's most recently computed assignment. At each step, the mini-batch's own rows are live (gradient-carrying); every other row is read from the memory as-is. Because a cell can go several steps between visits, we can optionally discount its remembered row by age — how long it has been since it last appeared in a mini-batch — so a long-unseen cell counts for less than one just computed (Appendix E gives the exact discount and why we leave it off in practice); a cell that reappears simply overwrites its old row with a fresh one. For multi-batch data, cohesion for a cell in batch b is restricted to $V(b)$, the cells either in b or directly connected to it, rather than all N cells (Appendix E).

Total Objective.

$$\mathcal{L} = \lambda_{\text{comm}}\,\mathcal{L}_{\text{community}} + \lambda_{\text{cvae}}\,\mathcal{L}_{\text{cVAE}} + \lambda_{\text{coh}}\,\mathcal{L}_{\text{cohesion}}, \tag{7}$$

where $\mathcal{L}_{\text{cVAE}}$ is the same per-cell reconstruction loss used to pretrain the encoder and decoder (Stage 1 below); we keep optimizing it during joint training, with the batch-embedding parameters learned in Stage 1 frozen. At this point we do not want the model to learn any new batch structure — the goal is for $\mathcal{L}_{\text{community}}$ and $\mathcal{L}_{\text{cohesion}}$ to shape the encoder and prototypes around correct *within-batch* similarity — but the decoder still needs z_i to carry enough information to reconstruct real gene expression, so $\mathcal{L}_{\text{cVAE}}$ stays in the objective specifically to preserve the cross-batch alignment Stage 1 already achieved, rather than to teach the encoder anything new about batches. In practice we include two instances of $\mathcal{L}_{\text{cohesion}}$, one for each choice of sim in Appendix E; every term above has its own weight; see Appendix D for training details.

Training Procedure. Stage 1: Pretraining. We pretrain the encoder and decoder using scPoli (De Donno et al., 2023), a cVAE-based architecture for single-cell data that encodes batch identity via trainable batch embeddings rather than fixed one-hot vectors, enabling effective batch effect correction. The reconstruction part of pretraining is a per-cell MSE loss on log-normalized gene expression:

$$\mathcal{L}_{\text{cVAE}} = \frac{1}{N}\sum_i \|x_i - g_\phi(z_i, b_i)\|_2^2, \tag{8}$$

where $z_i = f_\theta(x_i, b_i)$. Like any VAE, pretraining also includes a KL term pulling the encoder's approximate posterior toward a standard normal prior, weighted by scPoli's own default schedule. This stage trains the encoder and batch embeddings before prototypes are introduced (see Appendix D for architecture and hyperparameter details).

Stage 2: Prototype initialization and learning. Prototypes are initialized via waypoint initialization (Persad et al., 2023) from the pretrained encoder embeddings, with temperature τ calibrated before joint optimization (Appendix F); batch-embedding parameters from Stage 1 are frozen for the remainder of training (initialization and KL-term details in Appendix D).

This two-stage procedure stabilizes training and yields prototypes that preserve within-batch structure, align shared cell states across batches, and represent compact, distinct communities. Full hyperparameter details, loss weights, and guidance on selecting K are provided in Appendix D.

4 EXPERIMENTS

Datasets. We evaluate on three multi-batch single-cell RNA-seq datasets (Luecken et al., 2022): **Pancreas** (16,382 cells, 9 sequencing technologies used as batches, 14 cell types), **PBMC (immune)** (30k cells, 5 batches, 9 cell types), and a **Lung** dataset (30k cells, 16 batches, 17 cell types). For spatial evaluations, we use a **non-small cell lung cancer (NSCLC)** spatial transcriptomics dataset (Pentimalli et al., 2025). After preprocessing (see Appendix A), this yields 28,804 cells across 18 cell types and 9 niches. Niche labels are defined by clustering cells based on the local cell-type composition of their spatial neighborhood, providing a measure of microenvironmental context that is related to but distinct from the spatial affinity used during training.

Affinity construction. For transcriptomic experiments, cell–cell affinities are computed using an adaptive Gaussian kernel on PCA embeddings (Persad et al., 2023), which scales each cell's bandwidth to its local neighborhood density, ensuring rare cell types form well-connected communities despite occupying sparse regions; combined with scProto's cross-batch aggregation, this enables stable metacells for rare populations underrepresented in individual batches. For spatial experiments, we build a BANKSY-style affinity (Singhal et al., 2024): each cell's own PCA embedding is concatenated with a distance-weighted mean of its spatial neighbors' PCA embeddings (its spatial context), balanced by a spatial weight λ, and the same adaptive Gaussian kernel is then applied to the concatenated features (see Appendix C).

Baselines. We compare against three **metacell** methods — **SEACells** (Persad et al., 2023) (archetypal analysis), **MetaCell2** (Ben-Kiki et al., 2022) (divide-and-conquer partitioning), **SuperCell** (Bilous et al., 2022) (graph coarse-graining) — **MetaQ** (Li et al., 2025) (end-to-end vector-quantized, no batch correction), **scPoli (Stage-1)** (scProto's pretrained encoder before prototypes, Section 3), and three **batch-correction** methods — **Harmony** (Korsunsky et al., 2019), **BBKNN** (Polański et al., 2020), **ComBat** (Johnson et al., 2007). All methods use $d = 8$; scPoli (Stage-1) and the batch-correction methods lack native clusters, so we run Leiden (Traag et al., 2019) on their embedding, tuned to match scProto's K (the rest take K directly). For spatial experiments we add **SEACells (spatial)**: its default PCA affinity replaced by the spatial-context affinity.

Evaluation protocol. scProto outputs soft assignments $s_i \in \mathbb{R}^K$; cells take the argmax prototype and metacells are labeled by majority vote. We report **cohesion** (within-metacell similarity, Appendix E), **batch entropy**, **cell-type purity**, rare-cell **coverage/homogeneity**, and, for spatial data, **niche purity/DGE** (formal definitions in Appendix G). Table 1 additionally uses a **node-holdout** protocol: 20% of cells (stratified by batch) are excluded from the graph and training, with scProto/scPoli (Stage-1) placing them via a frozen-encoder forward pass, so those metrics are computed on never-trained-on cells only. Rare types are identified within each batch (frequency more than one std below the batch mean, else lowest quartile); scores are mean $\pm$ std across batches.

4.1 SCPROTO FORMS COHESIVE METACELLS WHILE INTEGRATING CELLS ACROSS BATCHES

As shown in Table 1, scProto matches the metacell baselines' cohesion (essentially tied with SEACells, the strongest of them) while clearly exceeding every batch-effect-removal method on it; batch entropy

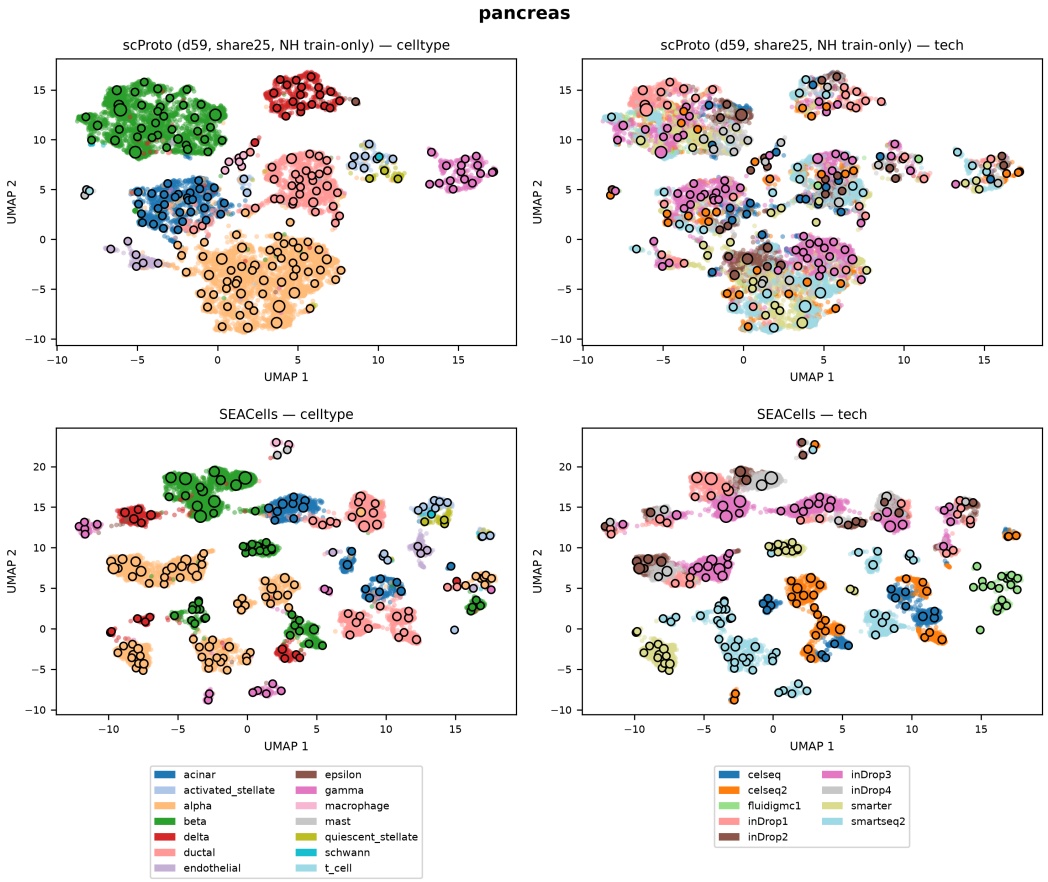


Figure 2: UMAP visualization of metacells on the Pancreas dataset (top: scProto, bottom: SEACells; left: cell type, right: batch). scProto forms compact, biologically coherent clusters while mixing cells across batches; SEACells produces batch-specific metacells. Circles with black outlines denote metacells (size $\propto$ assigned cells).

Table 1: Node-holdout protocol (metric definitions in §4): scProto/scPoli (Stage-1) place held-out cells via a frozen encoder; all values computed on held-out cells only. Other methods are fit on all cells (context, not matched). Bold: best; underlined: second-best.

	Pancreas				Immune				Lung			
Method	Coh(aff) ↑	Coh(gram) ↑	Entropy ↑	Purity ↑	Coh(aff) ↑	Coh(gram) ↑	Entropy ↑	Purity ↑	Coh(aff) ↑	Coh(gram) ↑	Entropy ↑	Purity ↑
scProto	**0.18 ± 0.13**	0.33 ± 0.19	1.21 ± 0.40	0.93 ± 0.12	0.08 ± 0.10	0.16 ± 0.16	0.70 ± 0.37	0.82 ± 0.16	0.13 ± 0.11	0.27 ± 0.17	1.16 ± 0.45	0.72 ± 0.19
scPoli (Stage-1) + Leiden	0.16 ± 0.12	0.28 ± 0.16	1.20 ± 0.46	0.97 ± 0.08	0.06 ± 0.06	0.11 ± 0.11	**0.88 ± 0.43**	0.87 ± 0.14	0.10 ± 0.10	0.19 ± 0.14	1.41 ± 0.59	0.84 ± 0.20
SEACells	0.18 ± 0.11	**0.36 ± 0.18**	0.16 ± 0.28	**0.97 ± 0.08**	**0.10 ± 0.10**	**0.21 ± 0.17**	0.15 ± 0.25	**0.88 ± 0.15**	**0.14 ± 0.10**	**0.30 ± 0.16**	0.38 ± 0.45	0.90 ± 0.15
BBKNN	0.11 ± 0.09	0.21 ± 0.14	0.83 ± 0.44	0.92 ± 0.12	0.05 ± 0.06	0.10 ± 0.10	0.51 ± 0.43	0.83 ± 0.15	0.07 ± 0.08	0.15 ± 0.12	1.18 ± 0.41	0.80 ± 0.20
Harmony	0.15 ± 0.11	0.25 ± 0.15	**1.24 ± 0.37**	0.91 ± 0.16	0.08 ± 0.06	0.14 ± 0.11	0.84 ± 0.45	0.84 ± 0.15	0.08 ± 0.09	0.15 ± 0.13	**1.49 ± 0.45**	0.72 ± 0.23
ComBat	0.17 ± 0.12	0.30 ± 0.16	0.82 ± 0.48	0.94 ± 0.13	0.06 ± 0.08	0.12 ± 0.13	0.42 ± 0.43	0.84 ± 0.15	0.10 ± 0.11	0.19 ± 0.17	1.00 ± 0.59	0.82 ± 0.19
MetaCell2	0.10 ± 0.13	0.19 ± 0.16	0.63 ± 0.59	0.88 ± 0.21	0.05 ± 0.10	0.10 ± 0.13	0.44 ± 0.44	0.80 ± 0.19	0.09 ± 0.07	0.20 ± 0.12	0.63 ± 0.54	0.90 ± 0.15
SuperCell	0.14 ± 0.12	0.27 ± 0.19	0.38 ± 0.43	0.93 ± 0.14	0.07 ± 0.10	0.14 ± 0.17	0.21 ± 0.28	0.86 ± 0.15	0.10 ± 0.09	0.21 ± 0.15	0.73 ± 0.50	0.90 ± 0.15
MetaQ	0.11 ± 0.07	0.24 ± 0.13	0.35 ± 0.41	0.96 ± 0.11	0.04 ± 0.04	0.08 ± 0.08	0.60 ± 0.51	0.81 ± 0.16	0.09 ± 0.07	0.21 ± 0.14	0.48 ± 0.50	**0.91 ± 0.13**

shows the reverse pattern, with scProto comparable to the batch-effect-removal methods and clearly ahead of every metacell method, which mix nothing across batches by construction (e.g. SEACells: 0.16/0.15/0.38 vs. scProto's 1.21/0.70/1.16). Cell-type purity stays within 0.06 of every other method on Pancreas and Immune, so cross-batch alignment does not cost biological identity there; Lung is the exception, where scProto's purity (0.72) trails every metacell baseline by 0.18–0.19, a real dataset-specific cost. Fig. 2 illustrates this: scProto's UMAP (McInnes et al., 2018) clusters align across batches while staying consistent with cell type, whereas SEACells' are largely batch-separated.

Table 2: Rare-cell representation, pooled over train + held-out cells (cf. Table 1's held-out-only protocol; metric definitions in §4). scProto/scPoli (Stage-1) use the node-holdout model; other methods are fit on the complete dataset (context, not matched). Bold: best; underlined: second-best.

	Pancreas			Immune			Lung		
Method	Homog ↑	F1 ↑	Coverage ↑	Homog ↑	F1 ↑	Coverage ↑	Homog ↑	F1 ↑	Coverage ↑
scProto	**0.48 ± 0.10**	**0.54 ± 0.11**	**0.77 ± 0.20**	0.83 ± 0.08	0.86 ± 0.02	**1.00 ± 0.00**	0.48 ± 0.19	0.55 ± 0.21	0.76 ± 0.28
scPoli (Stage-1) + Leiden	0.45 ± 0.20	0.33 ± 0.24	0.40 ± 0.24	0.78 ± 0.15	0.82 ± 0.16	1.00 ± 0.00	**0.57 ± 0.20**	0.62 ± 0.19	0.79 ± 0.25
SEACells	0.42 ± 0.20	0.37 ± 0.25	0.43 ± 0.27	**0.85 ± 0.07**	0.88 ± 0.08	1.00 ± 0.00	0.47 ± 0.24	0.42 ± 0.22	0.52 ± 0.23
BBKNN	0.16 ± 0.06	0.10 ± 0.08	0.17 ± 0.13	0.57 ± 0.23	0.60 ± 0.29	0.70 ± 0.27	0.36 ± 0.11	0.31 ± 0.17	0.45 ± 0.24
Harmony	0.22 ± 0.09	0.22 ± 0.17	0.40 ± 0.24	0.64 ± 0.13	0.75 ± 0.17	0.93 ± 0.13	0.34 ± 0.08	0.28 ± 0.14	0.43 ± 0.26
ComBat	0.23 ± 0.10	0.15 ± 0.07	0.33 ± 0.27	0.75 ± 0.18	0.80 ± 0.19	0.93 ± 0.13	0.40 ± 0.21	0.38 ± 0.19	0.53 ± 0.26
MetaCell2	0.20 ± 0.23	0.20 ± 0.26	0.26 ± 0.33	0.44 ± 0.28	0.52 ± 0.24	0.78 ± 0.19	0.56 ± 0.20	**0.64 ± 0.16**	**0.85 ± 0.20**
SuperCell	0.14 ± 0.06	0.07 ± 0.11	0.12 ± 0.18	0.84 ± 0.04	**0.88 ± 0.05**	1.00 ± 0.00	0.45 ± 0.16	0.47 ± 0.16	0.65 ± 0.22
MetaQ	0.20 ± 0.12	0.11 ± 0.14	0.25 ± 0.33	0.60 ± 0.12	0.73 ± 0.13	0.95 ± 0.10	0.54 ± 0.22	0.58 ± 0.23	0.70 ± 0.24

4.2 Cross-Batch Aggregation Improves Coverage; Cohesion Preserves Homogeneity

We examine whether density-corrected affinity combined with cross-batch aggregation improves representation of rare cell types.

As shown in Table 2, scProto's advantage is clearest against the batch-restricted metacell baselines (SEACells, SuperCell) on Pancreas and Lung, where a rare cell type often lacks enough cells within any single batch to form its own coherent metacell: on Pancreas, scProto leads both on homogeneity (0.48 vs. 0.42/0.14) and by a wide margin on F1 and coverage (0.54/0.77 vs. 0.37/0.43 for SEACells, 0.07/0.12 for SuperCell); on Lung the gap is smaller but in the same direction (homogeneity 0.48 vs. 0.47/0.45; F1 0.55 vs. 0.42/0.47; coverage 0.76 vs. 0.52/0.65). On Immune, where every batch already contains enough cells of each rare type for within-batch methods to reach full coverage, scProto is comparable rather than ahead: homogeneity (0.83), F1 (0.86), and coverage (1.00) all sit within 0.02 of SEACells and SuperCell's own scores. MetaCell2 and MetaQ follow a different pattern: both trail scProto clearly on Pancreas and Immune, but on Lung both score higher on homogeneity and F1 (MetaCell2: 0.56/0.64; MetaQ: 0.54/0.58; vs. scProto's 0.48/0.55); MetaCell2 also leads on Lung coverage (0.85 vs. scProto's 0.76), while MetaQ trails there (0.70).

Against the batch-correction baselines (scPoli-Stage-1+Leiden, Harmony, ComBat, BBKNN), scProto leads Harmony, ComBat, and BBKNN on every rare-cell metric on all three datasets — consistent with these methods correcting batch effects at the level of a whole embedding or cluster, which can absorb a rare, batch-thin population into a denser, spuriously similar neighbor before any rare-cell signal is scored. scPoli-Stage-1+Leiden is the exception: scProto still leads it on Pancreas (homogeneity 0.48 vs. 0.45, F1 0.54 vs. 0.33, coverage 0.77 vs. 0.40) and on Immune (homogeneity 0.83 vs. 0.78, F1 0.86 vs. 0.82, coverage tied at 1.00), but trails it on Lung (homogeneity 0.48 vs. 0.57, F1 0.55 vs. 0.62, coverage 0.76 vs. 0.79) — the one dataset where scProto's own pretrained encoder, clustered directly, out-recovers rare states relative to the full model.

4.3 scProto captures microenvironmental structure while preserving cell-type identity

We evaluate whether scProto's cell-type identity is grounded in gene expression itself rather than in whatever affinity graph it is given, unlike SEACells, which partitions whatever graph it is handed regardless of whether that structure is supported by gene expression. Using the BANKSY affinity's tunable spatial weight λ (Appendix C), we move the graph continuously from purely transcriptomic (λ=0) to increasingly spatially dominated (λ=0.2, 0.4, 0.8) and train both methods on the identical graph at each λ. As shown in Fig. 3, SEACells' cell-type purity collapses as λ rises ($0.82 \rightarrow 0.51$), and its niche-program recovery (Pearson r between metacell- and single-cell-level logFC) degrades similarly ($0.85 \rightarrow 0.66$), since its kernel is built directly from the BANKSY-mixed embedding and simply groups whatever the graph says is close; scProto degrades far less on both (purity $0.87 \rightarrow 0.74$, recovery $0.81 \rightarrow 0.74$), because its identity comes from an encoder that reads the cell's own gene expression directly, $z_i = f_\theta(x_i, b_i)$, trained via reconstruction to stay faithful to it—so the graph only shapes how prototypes group already expression-grounded encodings, rather than determining identity itself. The same pattern holds at the gene level: of the ground-truth niche-DE genes each method's metacells recover (size-weighted Welch t-test, $p < 0.05$, correct direction), scProto recovers **CXCL9**

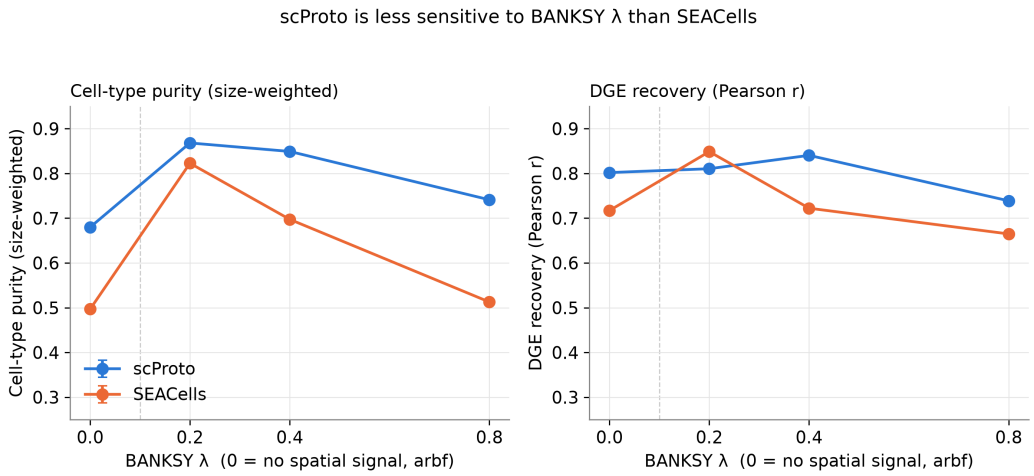


Figure 3: scProto vs. SEACells as the BANKSY spatial weight λ increases (0 = no spatial signal, using the adaptive-RBF affinity on PCA embeddings only). Left: size-weighted cell-type purity. Right: niche-program recovery (Pearson r between metacell-level and single-cell logFC, macro-averaged over cell-type/niche pairs). SEACells degrades sharply as λ rises; scProto stays comparatively flat.

within Fibroblasts at the Desmoplastic-stroma niche in 3 of 4 graphs while SEACells recovers it in none—a literature-consistent, IFNγ-induced chemokine marking the tumor–stromal interface and recruiting CXCR3$^+$ T cells (Chow et al., 2019), reported as one illustrative example rather than a systematic discovery claim.

5 Discussion and Conclusion

scProto addresses two core limitations of existing approaches by pairing an affinity-defined cohesion objective with a batch-conditioned reconstruction pretraining stage that establishes cross-batch alignment before prototypes are introduced: affinity-based methods preserve local structure but remain within-batch, limiting coverage of rare populations; reconstruction-based models align across batches but lose community structure, absorbing rare cell states into denser regions of the latent space. Two limitations remain: the number of prototypes K must be specified in advance and suboptimal choices can affect performance, and downstream quality depends on the affinity graph—noisy or poorly constructed affinities propagate misleading structure into the learned prototypes.

The framework extends naturally beyond transcriptomics: spatial affinities enable scProto to capture niche-correlated transcriptional programs while the reconstruction objective preserves cell-type identity. Future directions include more principled spatial affinities based on ligand–receptor interactions or local cell-type composition, and affinity-guided prototypes as compact, interpretable units within single-cell foundation models (Szałata et al., 2024; Richter et al., 2025; Bahrami et al., 2025)—aggregating pretrained embeddings while preserving local structure and complementing global alignment in multimodal settings (Richter et al., 2026; Bahrami et al., 2026).

AI use statement

Generative AI tools were used in this work in three capacities. First, for writing support: copy-editing and rephrasing of author-written text. Second, for coding and execution support: implementing and refactoring experiment and analysis scripts, executing the associated training and evaluation runs, and reading the resulting output files to report and tabulate the numerical values used in this paper. Third, for limited mathematical assistance: helping derive and explore some formula and normalization details of the method during implementation. All experimental design decisions — which experiments to run, which configurations and hyperparameters to use, and which results to report — were made by the authors; generative AI tools were not used to decide the experimental

design or to select which results to include. All AI-executed code, all AI-assisted mathematical derivations, and all AI-reported experimental results were reviewed and verified by the authors against the underlying implementation and output files, and all AI-assisted text was reviewed and edited by the authors. We take responsibility for the final content of this work, including all text, claims, experimental results, and artifacts.

ETHICS STATEMENT

This work uses only publicly available, de-identified single-cell and spatial transcriptomics datasets, obtained and used in accordance with their original licences and data-use terms; no new human or animal data were collected and no IRB approval was required. The method produces aggregated cell profiles (metacells) for exploratory analysis and is not intended for clinical decision making. The authors declare no conflicts of interest.

REPRODUCIBILITY STATEMENT

The method, objective, and all loss terms are specified in Section 3. Dataset sources and preprocessing are described in Appendix A, affinity-graph construction in Appendix B, and architecture, hyperparameters, and the training schedule in the training appendix. Evaluation metrics are defined with explicit formulas where they are introduced, and all reported numbers are given with standard deviations. Code to reproduce every table and figure will be made publicly available upon acceptance.

REFERENCES

Gabriel Y Arteaga et al. Why prototypes collapse: Diagnosing and preventing partial collapse in prototypical self-supervised learning. In *International Conference on Learning Representations*, 2026.

Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi. Self-labelling via simultaneous clustering and representation learning. *arXiv preprint arXiv:1911.05371*, 2019.

Mojtaba Bahrami, Alejandro Tejada-Lapuerta, Sören Becker, Fatemeh S. Hashemi G., and Fabian J. Theis. scconcept: Contrastive pretraining for technology-agnostic single-cell representations beyond reconstruction. *bioRxiv*, 2025. doi: 10.1101/2025.10.14.682419.

Mojtaba Bahrami, Till Richter, Nico A. Schmacke, Alejandro Egea Lavandera, and Fabian J. Theis. From modality-specific to compositional foundation models for cell biology. *Cell Systems*, 17(2): 101534, 2026. doi: 10.1016/j.cels.2026.101534.

Y. Baran et al. Metacell: analysis of single-cell rna-seq data using k-nn graph partitions. *Genome Biology*, 2019. https://doi.org/10.1186/s13059-019-1812-2.

O. Ben-Kiki, A. Bercovich, A. Lifshitz, and A. Tanay. Metacell-2: a divide-and-conquer metacell algorithm for scalable scrna-seq analysis. *Genome Biology*, 23(1):100, 2022.

Mariia Bilous, Loc Tran, Chiara Cianciaruso, Aurélie Gabriel, Hugo Michel, Santiago J. Carmona, Mikael J. Pittet, and David Gfeller. Metacells untangle large and complex single-cell transcriptome networks. *BMC Bioinformatics*, 23:336, 2022. doi: 10.1186/s12859-022-04861-1.

V. D. Blondel et al. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008. https://doi.org/10.1088/1742-5468/2008/10/P10008.

Robin Browaeys, Wouter Saelens, and Yvan Saeys. Nichenet: modeling intercellular communication by linking ligands to target genes. *Nature Methods*, 17:159–162, 2020. doi: 10.1038/s41592-019-0667-5.

Z. Cai, Z. Hu, S. Sun, Z. Wang, F. Yang, J. Han, and F. Zeng. Distribution-preserved compression of single-cell atlases for privacy-protected data dissemination and novel cell type discovery. *bioRxiv*, 2024. doi: 10.1101/2024.11.16.622584.

M. Caron et al. Unsupervised learning of visual features by contrasting cluster assignments. In *NeurIPS*, 2020. https://arxiv.org/abs/2006.09882.

Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *European Conference on Computer Vision (ECCV)*, pp. 139–156, 2018. doi: 10.1007/978-3-030-01264-9_9.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International Conference on Machine Learning*, pp. 794–803, 2018.

Melvyn T Chow, Aleksandra J Ozga, Rachel L Servis, Dennie T Frederick, Jonathan A Lo, David E Fisher, Gordon J Freeman, Genevieve M Boland, and Andrew D Luster. Intratumoral activity of the CXCR3 chemokine system is required for the efficacy of anti-PD-1 therapy. *Immunity*, 50(6): 1498–1512, 2019. doi: 10.1016/j.immuni.2019.04.010.

Adele Cutler and Leo Breiman. Archetypal analysis. *Technometrics*, 36(4):338–347, 1994. doi: 10.1080/00401706.1994.10485840.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in Neural Information Processing Systems*, 26:2292–2300, 2013.

Carlo De Donno, Soroor Hediyeh-Zadeh, Amir Ali Moinfar, Marco Wagenstetter, Luke Zappia, Mohammad Lotfollahi, and Fabian J. Theis. Population-level integration of single-cell datasets enables multi-scale analysis across samples. *Nature Methods*, 20:1683–1692, 2023. doi: 10.1038/s41592-023-02035-2.

Inderjit S Dhillon, Yuqiang Guan, and Brian Kulis. Kernel k-means, spectral clustering and normalized cuts. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 551–556, 2004.

Fatemeh S. Hashemi Golpayegani, Till Richter, Alejandro Tejada Lapuerta, Lennard Halle, Mohammad Lotfollahi, and Fabian J. Theis. Interpretable self-supervised prototype learning for single-cell transcriptomics. In *Learning Meaningful Representations of Life (LMRL) Workshop at the International Conference on Learning Representations (ICLR)*, 2025. URL `https://openreview.net/forum?id=mTjWUeyll5`.

Suoqin Jin, Christian F Guerrero-Juarez, Lihua Zhang, Ivan Chang, Raul Ramos, Chun-Hsiang Kuan, Peggy Myung, Maksim V Plikus, and Qing Nie. Inference and analysis of cell-cell communication using cellchat. *Nature Communications*, 12:1088, 2021. doi: 10.1038/s41467-021-21246-9.

W. Evan Johnson, Cheng Li, and Ariel Rabinovic. Adjusting batch effects in microarray expression data using empirical bayes methods. *Biostatistics*, 8(1):118–127, 2007. doi: 10.1093/biostatistics/kxj037.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

I. Korsunsky et al. Fast, sensitive and accurate integration of single-cell data with harmony. *Nature Methods*, 2019. https://doi.org/10.1038/s41592-019-0619-0.

Y. Li et al. Metaq: fast, scalable and accurate metacell inference via single-cell quantization. *Nature Communications*, 16(1):1205, 2025.

Qi Liu, Yu Wang, Chih-Yuan Hsu, Celestine N Wanjalla, Ken S Lau, and Yu Shyr. Microenvironment-aware spatial modeling for accurate inference of cell identity. *Nucleic Acids Research*, 54(1), 2026. doi: 10.1093/nar/gkaf1477.

R. Lopez et al. scvi: Deep generative modeling for single-cell transcriptomics. *Nature Methods*, 2018. https://doi.org/10.1038/s41592-018-0229-2.

Malte Luecken, Maren Büttner, Anna Danese, Marta Interlandi, Michaela Müller, Daniel Strobl, Luke Zappia, Martin Dugas, Maria Colomé-Tatché, Fabian Theis, and Kridsadakorn Chaichoompu. Benchmarking atlas-level data integration in single-cell genomics – integration task datasets, 2022. Dataset. `https://doi.org/10.6084/m9.figshare.12420968.v8`.

Malte D Luecken and Fabian J Theis. Current best practices in single-cell rna-seq analysis: a tutorial. *Molecular systems biology*, 15(6):e8746, 2019.

Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

Tancredi Massimo Pentimalli, Simon Schallenberg, Daniel León-Periñán, Ivano Legnini, Ilan Theurillat, Gwendolin Thomas, Anastasiya Boltengagen, Sonja Fritzsche, Jose Nimo, Lukas Ruff, et al. Combining spatial transcriptomics and ecm imaging in 3d for mapping cellular interactions in the tumor microenvironment. *Cell systems*, 16(5), 2025.

Sitara Persad, Zi-Ning Choo, Christine Dien, Noor Sohail, Ignas Masilionis, Ronan Chaligné, Tal Nawy, Chrysothemis C. Brown, Roshan Sharma, Itsik Pe'er, Manu Setty, and Dana Pe'er. Seacells infers transcriptional and epigenomic cellular states from single-cell genomics data. *Nature Biotechnology*, 41:1746–1757, 2023. doi: 10.1038/s41587-023-01716-9.

Krzysztof Polański, Matthew D. Young, Zhichao Miao, Kerstin B. Meyer, Sarah A. Teichmann, and Jong-Eun Park. Bbknn: fast batch alignment of single cell transcriptomes. *Bioinformatics*, 36(3): 964–965, 2020. doi: 10.1093/bioinformatics/btz625.

Till Richter, Mojtaba Bahrami, Yufan Xia, et al. Delineating the effective use of self-supervised learning in single-cell genomics. *Nature Machine Intelligence*, 7:68–78, 2025. doi: 10.1038/s42256-024-00934-3.

Till Richter, Eric Zimmermann, James Hall, Fabian J. Theis, Srivatsan Raghavan, Peter S. Winter, Ava Pardis Amini, and Lorin Crawford. Beyond alignment: synergistic integration is required for multimodal cell foundation models. *bioRxiv*, 2026. doi: 10.64898/2026.02.23.707420.

Tim Sainburg, Leland McInnes, and Timothy Q. Gentner. Parametric umap embeddings for representation and semisupervised learning. *Neural Computation*, 33(11):2881–2907, 2021. doi: 10.1162/neco_a_01434.

Vipul Singhal, Nigel Chou, Joseph Lee, Yifei Yue, Jinyue Liu, Wan Kee Chock, Li Lin, Yun-Ching Chang, Erica Mei Ling Teo, Jonathan Aow, et al. BANKSY unifies cell typing and tissue domain segmentation for scalable spatial omics data analysis. *Nature Genetics*, 56:431–441, 2024. doi: 10.1038/s41588-024-01664-3.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems*, volume 28, pp. 3483–3491, 2015.

Artur Szałata et al. Transformers in single-cell omics: a review and new perspectives. *Nature Methods*, 21(8):1430–1443, 2024. doi: 10.1038/s41592-024-02353-z.

Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Addison-Wesley, 2005.

Vincent A Traag, Ludo Waltman, and Nees Jan van Eck. From louvain to leiden: guaranteeing well-connected communities. *Scientific reports*, 9(1):5233, 2019.

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Advances in neural information processing systems*, volume 30, 2017.

F Alexander Wolf, Philipp Angerer, and Fabian J Theis. Scanpy: large-scale single-cell gene expression data analysis. *Genome biology*, 19(1):15, 2018.

Hongkui Zeng. What is a cell type and how to define it? *Cell*, 185(15):2739–2755, 2022.

A Dataset Preprocessing Details

The three scRNA-seq datasets (Pancreas, PBMC, Lung) are taken from the benchmarking collection of Luecken et al. (2022). For the NSCLC spatial transcriptomics dataset (Pentimalli et al., 2025), the original data comprises 6 slides; we select slide 28, subsample 30,000 cells stratified by cell type and niche label, and exclude cells assigned to uninformative niche labels, yielding 28,804 cells across 18 cell types and 9 niches. Niche labels are defined by clustering cells based on the local cell-type composition of their spatial neighborhood. All datasets are preprocessed with Scanpy (Wolf et al., 2018) by normalizing total counts and applying log1p transformation. For the scRNA-seq datasets, 4,000 highly variable genes are selected; for NSCLC, HVG selection is omitted as the panel comprises approximately 950 genes.

B Affinity Graph and Batch Structure

The affinity graph P is computed using an adaptive Gaussian (RBF) kernel in PCA space, following the SEACells protocol (Persad et al., 2023) with default hyperparameter choices (15 nearest neighbors for bandwidth estimation). Because PCA embeddings reflect both biological variation and technical batch effects, the resulting graph encodes a mixture of biological and batch-driven proximity. Figure 4 visualizes this directly: a UMAP embedding of the affinity matrix on the Pancreas dataset (Luecken et al., 2022) reveals clear batch structure alongside biological cell-type organization. This confirms that the input graph is not batch-free, and motivates the batch-conditional encoder design, which explicitly disentangles these signals during training rather than requiring a pre-corrected graph.

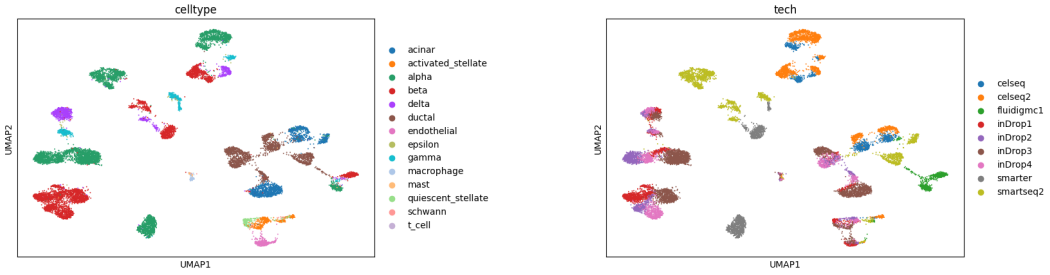


Figure 4: UMAP of the precomputed RBF affinity graph on the Pancreas dataset. Left: colored by cell type. Right: colored by batch label. Batch structure is clearly visible in the graph embedding, confirming that the input affinity graph reflects both biological and technical variation. The batch-conditional encoder disentangles these signals during training.

C Spatial Affinity Construction

For spatial experiments, the cell–cell affinity is built with a BANKSY-style feature augmentation (Singhal et al., 2024) rather than from individual cell transcriptomes alone. Concretely, for each cell i we compute a spatial context vector $\tilde{x}_i$, the distance-weighted mean of its spatial neighbors' PCA embeddings, and concatenate it with the cell's own PCA embedding:

$$x_i^{\text{aug}} = \left[\sqrt{1-\lambda^2} \cdot \text{PCA}(x_i),\ \ \lambda \cdot \tilde{x}_i \right], \tag{9}$$

where the spatial weight $\lambda \in [0, 1]$ sets the target fraction of feature variance contributed by spatial context; the two concatenation coefficients are then derived from λ together with the empirical total variance of each block (V_{pca}, the variance of the PCA block, and V_{ctx}, the variance of the context block), so that the realized variance split matches λ regardless of the two blocks' raw scales. The same adaptive Gaussian (RBF) kernel used for transcriptomic affinity is then applied to x_i^{aug} to construct the spatial affinity graph. At $\lambda = 0$ this reduces exactly to the transcriptomic-only affinity; Section 4 sweeps $\lambda \in \{0, 0.2, 0.4, 0.8\}$.

As λ increases, affinity increasingly reflects shared local microenvironment over individual transcriptomic similarity, while the cell's own expression (the first block) still contributes at every $\lambda > 0$.

Because the model receives only the cell's own gene expression as input, the community-preserving objective forces the encoder to identify gene expression patterns that correlate with microenvironmental context, while the reconstruction objective keeps the encoder's latent space z faithful to overall gene expression, preventing spatial context from overriding cell-type identity there. Prototypes, which the cohesion objective places among real cells in that same z-space (main text), inherit this grounding rather than being anchored to gene expression directly.

D Training Details and Hyperparameters

Prototype initialization (Stage 2). Waypoint initialization (Persad et al., 2023) selects K cells by greedy MaxMin in diffusion map space and uses their pretrained encoder embeddings as initial prototype vectors. From Stage 2 onward we drop the KL term, since Stage-1 pretraining already regularized the latent space.

Why community cannot be allowed to move the prototypes. We measured what happens if $\mathcal{L}_{\text{community}}$ is allowed to update $\{c_k\}$ directly, i.e. using the live s_i rather than the frozen $\hat{s}_i$ in q_{ij} (main text): at full strength, this loss's negative-sampling repulsion accounted for 89% of the gradient reaching the prototypes and killed most of them (14–16 of 200 dead in our runs). This is a structural property of the objective, not a tuning artifact: negative pairs push dissimilar cells' prototypes apart far more strongly than positive pairs pull similar cells' prototypes together, so while this objective's optimum for the *encoder* is exactly what we want, its optimum for the *prototypes* is degenerate. Routing q_{ij} through a stop-gradient copy of the prototypes removes this failure mode while leaving the encoder signal untouched.

Architecture. The encoder and decoder follow the scPoli architecture (De Donno et al., 2023) with default settings: hidden dimension 64, latent dimension 8, and 50 PCA components as input features.

ℓ_2 normalization. Following Caron et al. (2020), both cell embeddings z_i and prototype vectors c_k are projected onto the unit hypersphere before computing soft assignments, so the dot product $z_i^\top c_k$ is bounded in $[-1, 1]$. Prototype vectors are re-normalized at the start of each mini-batch, immediately after the preceding gradient update.

Optimizer. We use Adam (Kingma & Ba, 2014) with learning rate 3×10^{-4}, $\varepsilon = 0.01$, and weight decay 10^{-5}. Batch embedding parameters are excluded from weight decay to avoid penalizing dataset-specific offsets. A linear warmup over 5 epochs is followed by cosine learning rate decay.

Batch size. Mini-batches contain 1024 cells sampled uniformly across batches.

Training stages. Stage 1 (cVAE pretraining) runs for 50 epochs. Stage 2 (joint prototype learning) uses early stopping based on modularity, evaluated every few epochs, with a minimum improvement threshold of 0.005 and 1000 gradient steps per epoch.

Loss weights. The total objective (main text, and Appendix E for the two cohesion similarities) has four active terms: community, cVAE reconstruction, and the two cohesion variants (affinity-based and Gram-based). λ_{coh} (for each cohesion variant) is set directly from the training graph: we measure a *span* — the gap between a well-clustered partition's cohesion score and a random partition's, on that graph — and set $\lambda = \text{authority}/\text{span}$, where authority is a fixed reference value carried across datasets and span is measured per dataset, so the same relative emphasis on cohesion is preserved even though graphs differ substantially in scale.

λ_{comm} and λ_{cvae} are tuned per dataset. We first measure each term's own encoder-gradient norm on a single mini-batch at the start of training (temporarily setting its weight to 1), which anchors a sensible numeric range for that term on that dataset's graph, since raw gradient scales are not portable across graphs (Chen et al., 2018). Within that range, we run a short sweep and select the weight that maximizes held-out purity subject to a hard constraint: every prototype must remain populated (zero dead prototypes) throughout training.

Gradient routing summary. Each of the three joint-training terms is deliberately routed to a different subset of parameters, so that no two terms compete for the same update:

$$\begin{array}{lcl} \mathcal{L}_{\text{community}} & \rightarrow & f_\theta \text{ only (prototypes detached, main text)} \\ \mathcal{L}_{\text{cohesion}} & \rightarrow & f_\theta \text{ and } \{c_k\} \text{ (live assignments, main text)} \\ \mathcal{L}_{\text{cVAE}} & \rightarrow & f_\theta \text{ and } g_\phi \text{ (no prototype involved at all)} \end{array} \tag{10}$$

$\mathcal{L}_{\text{cohesion}}$ is therefore the only term that shapes $\{c_k\}$ directly; $\mathcal{L}_{\text{community}}$ and $\mathcal{L}_{\text{cVAE}}$ both shape the encoder, from two different signals (affinity-graph structure and decodability, respectively), and never see the prototypes. A prototype's own gene-expression profile, when one is needed for interpretation, is read out by decoding c_k through the already-trained decoder, $g_\phi(c_k, b)$ — this is a post-hoc readout of a shared decoder trained entirely on real cells, not a training objective of its own, and no loss term ever compares its output against anything.

Number of prototypes K. Following SEACells (Persad et al., 2023), we set $K \approx N/75$, where N is the total number of cells. This yields approximately one metacell per 75 cells and provides a natural starting point for unannotated datasets. The ratio of 75 cells per metacell reflects a target aggregation granularity that preserves rare populations while achieving meaningful compression. Users may increase K for finer resolution or decrease it for coarser summaries.

Graph construction. The affinity graph uses an adaptive Gaussian (RBF) kernel in 50-dimensional PCA space, with 15 nearest neighbors used to estimate per-cell bandwidth, following the SEACells protocol (Persad et al., 2023). All graph hyperparameters are kept at their SEACells defaults across all experiments. For spatial experiments, see Appendix C.

E Cohesion: Similarity Choices and Assignment Memory

Two choices of similarity. The cohesion objective (main text) is defined for a generic cell–cell similarity $\mathrm{sim}(i, j)$. We use two concrete instantiations simultaneously, each as its own copy of $\mathcal{L}_{\text{cohesion}}$ with its own weight:

Direct affinity. $\mathrm{sim}_{\text{aff}}(i, j) = P_{ij}$, the raw affinity graph itself. This is a *1-hop* notion of similarity: it asks whether i and j are directly connected in the graph. Under this choice, $\mathcal{L}_{\text{cohesion}}$ reduces to a ratio-association objective: for each prototype, the average affinity among its own members.

Gram similarity. $\mathrm{sim}_{\text{gram}}(i, j) = \cos(v_i, v_j)$, the cosine similarity between rows i and j of the affinity graph, treating each row $P_{i,:}$ as a vector of i's connections to every other cell. This is a *2-hop* notion — *structural equivalence*: two cells score high not because they are connected to each other, but because they are connected to the *same* other cells, i.e. they occupy the same neighborhood. Two cells can be structurally equivalent without being adjacent at all (e.g. two cells on opposite sides of the same hub), which is exactly the case the direct-affinity similarity above cannot see, and the two are combined for that reason: proximity ($\mathrm{sim}_{\text{aff}}$) and structural equivalence ($\mathrm{sim}_{\text{gram}}$) are a classical non-redundant pair, and neither alone catches the failure mode of the other.

One subtlety in constructing v_i: P has a zero diagonal by convention (self-loops are removed for the graph-cut style computations elsewhere in the pipeline), even though the affinity kernel itself would naturally assign $P_{ii} = 1$. Computing the cosine directly on these zero-diagonal rows makes two cells that are directly adjacent, but otherwise share no neighbors, score *exactly* 0 — an artifact of the missing self-loop, not a real absence of similarity. We correct this by closing the neighborhood before normalizing: $v_i = P_{i,:} + \delta\, e_i$ (adding back a self-weight δ on the diagonal), then $\mathrm{sim}_{\text{gram}}(i, j) = v_i^\top v_j / (\|v_i\|\, \|v_j\|)$. We use $\delta = 1$, matching the kernel's natural self-similarity and giving $\mathrm{sim}_{\text{gram}}(i, i) = 1$, consistent with the $\mathrm{sim}(i, i) := 1$ convention used throughout. This is also not a new convention we invented: SEACells (Persad et al., 2023) sets the diagonal of its own kNN graph to 1 explicitly, to "include self as a neighbour," and its RBF kernel gives $\exp(0) = 1$ on the diagonal by construction — we are matching an existing, working convention, not introducing one.

Why $\mathrm{sim}(i, i) := 1$ everywhere. This convention is not only used inside the Gram construction above — it applies to every similarity we use, and it is what avoids dead prototypes and lets training use the full K slots (main text). An empty prototype scores exactly 0. Without fixing $\mathrm{sim}(i, i) := 1$, a

prototype that has only just recruited its first member would score 0 too — its founding member's only pair is itself, and that pair would contribute nothing — so the objective could not tell a barely-started prototype from an empty one, and would never reward the model for growing it further. Fixing $\mathrm{sim}(i, i) := 1$ gives that prototype a score of 1 instead, enough of a foothold to keep recruiting rather than being abandoned.

Assignment memory. As described in the main text, cohesion needs a full-dataset size_k and $\overline{\mathrm{sim}}(i, k)$, so we keep a memory S^{mem} of every cell's last-computed assignment and overwrite only the current mini-batch's rows at each step. Two further details:

Initialization. Before the first step, S^{mem} can either start uniform ($1/K$ in every entry, carrying no information until a cell's first visit) or be filled once with a real forward pass of the pretrained encoder over all N cells. We use the latter, so that prototype sizes and similarities are informative from step 0 rather than only after a full epoch of mini-batches has passed through the memory.

Age discount (optional). A memory row is only as fresh as the last mini-batch that contained that cell, so far-apart revisits mean some rows are stale. We can down-weight a row by $0.5^{\,\Delta t/h}$, where Δt is the number of steps since that cell was last seen and h is a half-life, applied identically to both the numerator ($\overline{\mathrm{sim}}$) and denominator (size_k) so that the discount does not bias the ratio. In our runs, the measured drift between a cell's remembered and freshly recomputed assignment across a typical revisit interval was small, indicating the discount is unnecessary at our batch sizes and dataset scales; we report this as a diagnostic rather than always enabling it.

Batch restriction. The affinity graph P is not exactly block-diagonal by batch — some cross-batch edges exist — but cross-batch similarity is dominated by the batch effect rather than real biological similarity (Appendix B), so we do not want a cross-batch *absence* of an edge to count as evidence of dissimilarity either way. We handle this by scoring a cell in batch b only against $V(b)$, the set of cells that are either in b or directly connected to it, and restrict *both* size_k and the numerator of $\overline{\mathrm{sim}}(i, k)$ to $V(b)$ when computing cohesion for that cell — not just one of the two. Restricting only the numerator (scoring against $V(b)$ but dividing by the true, dataset-wide size_k) would systematically score a batch-pure prototype far below an otherwise identical batch-mixed one, purely because a batch-pure prototype's members are a strict subset of what a batch-mixed prototype's members would be under the same global denominator; restricting both sides removes this bias. With a single batch, $V(b)$ is every cell and this reduces to the unrestricted formula exactly.

F Temperature Calibration

The temperature τ in the soft assignment formula controls the sharpness of prototype assignments: a smaller τ yields near-one-hot assignments concentrated on a single prototype, while a larger τ spreads probability across many prototypes. Rather than setting τ as a fixed hyperparameter, we calibrate it from data after encoder pretraining and prototype initialization.

Effective number of prototypes. We characterize assignment sharpness through the effective number of prototypes per cell,

$$\mathrm{effk}_i = \frac{1}{\sum_k s_{ik}^2}, \tag{11}$$

which equals 1 when a cell is assigned entirely to one prototype and K when assignments are perfectly uniform.

Default calibration: affinity-matching. For dot-product similarity ($q_{ij} = s_i^\top s_j$), we calibrate τ so that the expected assignment similarity on positive edges matches the expected affinity weight:

$$\mathbb{E}_{(i,j)\sim P}\left[s_i^\top s_j\right] = \mathbb{E}_{(i,j)\sim P}\left[P_{ij}\right]. \tag{12}$$

Intuitively, this sets the temperature so that the degree of overlap between prototype assignments on truly connected cell pairs mirrors the strength of their affinity. We solve for τ via binary search, holding the encoder logits fixed.

If the target is unreachable (i.e., even at the sharpest feasible τ the left-hand side cannot reach the right-hand side), or if the resulting mean effective number of prototypes falls below 3 (indicating over-collapse), we fall back to effective-k alignment.

Fallback: effective-k alignment. We binary-search for τ such that the mean effective number of prototypes per cell equals a target K_{eff}:

$$\text{mean}_i \, \text{effk}_i = K_{\text{eff}}. \tag{13}$$

The target K_{eff} is derived from the median effective number of neighbors in the affinity graph, $\text{effk}_i^{\text{aff}} = 1/\sum_j P_{ij}^2$. This alignment ensures that assignment sharpness is commensurate with the local neighborhood structure of the data. This fallback is also the default for non-dot-product similarity metrics, for which the affinity-matching criterion is not meaningful.

Calibration runs once on the full dataset immediately after prototype initialization and before joint optimization begins.

G Evaluation Metrics

Cohesion. For a cell i assigned to metacell $m(i)$, cohesion is i's mean similarity to the other members of $m(i)$ under a chosen cell–cell similarity $\text{sim}(\cdot, \cdot)$, with $\text{sim}(i, i) := 1$ by the same convention used in training (main text):

$$\text{coh}(i) = \frac{1}{|m(i)|} \sum_{j \in m(i)} \text{sim}(i, j). \tag{14}$$

We report this under the two similarity choices used in training (Appendix E): **Coh(aff)**, using the affinity graph directly, and **Coh(gram)**, using the Gram/structural-equivalence similarity. Under the node-holdout protocol (Table 1), cohesion is computed only for held-out cells, against the full (train + held-out) affinity graph, with mean and standard deviation reported across held-out cells; the multi-batch restriction to $V(b)$ (Appendix E) applies identically at evaluation time.

Batch entropy. For each metacell m, batch entropy is the Shannon entropy of its batch label distribution:

$$H(m) = -\sum_b p_b \log p_b, \tag{15}$$

where p_b is the fraction of cells in m from batch b. Higher values indicate more uniform mixing across batches.

Cell-type purity. Purity measures the fraction of cells in a metacell belonging to its most frequent cell type:

$$\text{purity}(m) = \frac{1}{|m|} \max_c \sum_{i \in m} \mathbf{1}[y_i = c]. \tag{16}$$

Rare cell type metrics. All rare-cell metrics identify rare cell types *within each batch separately*. Let $f_c^{(b)}$ be the frequency of cell type c in batch b (fraction of cells). A cell type is locally rare in b if $f_c^{(b)} < \bar{f}^{(b)} - \sigma_f^{(b)}$, where $\bar{f}^{(b)}$ and $\sigma_f^{(b)}$ are the mean and standard deviation of within-batch cell-type frequencies. If no cell type satisfies this criterion, we fall back to the 25th percentile of $\{f_c^{(b)}\}$. All per-batch scores are aggregated as mean $\pm$ std across batches.

Coverage (per batch) is the fraction of locally-rare cell types for which at least one cell of that type in the current batch is assigned to a metacell whose majority label is that cell type.

For each cell i with a locally-rare label c in batch b, the **homogeneity score** is the fraction of cells in i's metacell $m(i)$ that share label c (across all batches):

$$h_i = \frac{1}{|m(i)|} \sum_{j \in m(i)} \mathbf{1}[y_j = c]. \tag{17}$$

The per-batch **homogeneity** is the mean of h_i over all locally-rare cells in batch b. Unlike coverage, which only checks whether a dedicated metacell exists, homogeneity measures how coherently each rare cell is grouped—a method can achieve full coverage while assigning rare cells to large, impure metacells, which homogeneity captures.

For each locally-rare type c in batch b, define its *recall* as the fraction of c's cells in batch b that sit in a metacell whose majority label is c, and its *precision* as the purity of the metacells majority-labelled c, size-weighted, computed once over the *complete* dataset (not per batch) since a type's dedicated metacells can draw members from every batch:

$$\mathrm{recall}(c,b) = \frac{1}{|\{i : y_i = c,\, b_i = b\}|} \sum_{i:\, y_i=c,\, b_i=b} \mathbf{1}\big[\mathrm{majority}(m(i)) = c\big], \qquad \mathrm{precision}(c) = \sum_{m:\, \mathrm{majority}(m)=c} \frac{|m|}{\sum_{m':\, \mathrm{majority}(m} \tag{18}$$

F1 for batch b is the harmonic mean of precision and recall, averaged over b's locally-rare types:

$$\mathrm{F1}(b) = \frac{1}{|\mathrm{rare}(b)|} \sum_{c \in \mathrm{rare}(b)} \frac{2\,\mathrm{precision}(c)\,\mathrm{recall}(c,b)}{\mathrm{precision}(c) + \mathrm{recall}(c,b)}. \tag{19}$$

Recall is batch-local; precision is not — a type's precision is the same number in every batch where it appears as locally rare, since it depends only on that type's dedicated metacells dataset-wide, not on which batch is being scored.

Niche purity. Analogous to cell-type purity but computed with respect to spatial niche labels: each metacell is assigned a majority niche label, and niche purity is the fraction of its member cells sharing that label. Unlike cell-type purity and batch entropy, niche purity is computed without size weighting, treating each metacell as an equal spatial unit. This is appropriate because niche-driven expression changes can be subtle, and the metacells capturing them tend to be small; size-weighting would suppress exactly these rare and spatially specific structures.

Niche-program recovery (DGE). While niche purity checks whether a metacell's members share a niche label, it says nothing about whether the metacell's *expression profile* recovers the transcriptional program that actually defines that niche. We measure this directly against a single-cell ground truth. For each (cell type, niche) pair, we first establish ground-truth differential expression: within that cell type's real single cells, a niche-vs-rest Wilcoxon test gives every gene a single-cell log fold-change $\mathrm{logFC}_g^{\mathrm{sc}}$; we keep genes significant at $\mathrm{padj} < 0.05$ (top 100 by $|\mathrm{logFC}^{\mathrm{sc}}|$). On the metacell side, we build one pseudobulk profile per metacell by summing real member cells' raw counts (CPM-normalized), then split metacells of the target cell type into *positive* (majority niche label matches) and *control* (it does not). The metacell-level fold-change for each ground-truth gene g is

$$\mathrm{logFC}_g^{\mathrm{mc}} = \log_2\left(\frac{\overline{\mathrm{cpm}}_g^{\mathrm{pos}} + 1}{\overline{\mathrm{cpm}}_g^{\mathrm{ctrl}} + 1}\right), \tag{20}$$

where $\overline{\mathrm{cpm}}_g^{\mathrm{pos}}$ and $\overline{\mathrm{cpm}}_g^{\mathrm{ctrl}}$ are gene g's mean CPM across positive and control metacells, respectively. **DGE** for a (cell type, niche) pair is the Pearson correlation between $\mathrm{logFC}^{\mathrm{mc}}$ and $\mathrm{logFC}^{\mathrm{sc}}$ across the ground-truth genes; we report it macro-averaged, first across niches within a cell type, then across cell types — the same two-level averaging used for purity. A pair is only scored if it clears a minimum-support gate (at least 2 positive and 2 control metacells, at least 5 ground-truth genes); Fig. 3 (main text) reports this average as the spatial weight λ varies.