MaskGraphene: Advancing joint embedding, clustering, and batch correction for spatial transcriptomics using graph-based self-supervised learning

Yunfei Hu[1], Yikang Li[2], Manfei Xie[2], Mingxing Rao[1], Yichen Henry Liu[1], and Xin Maizie Zhou[1,2,*]

[1] Department of Computer Science, Vanderbilt University, Nashville, USA
[2] Department of Biomedical Engineering, Vanderbilt University, Nashville, USA
maizie.zhou@vanderbilt.edu

Abstract. With the rapid advancement and generation of spatial transcriptomics (ST), integrating data from multiple ST slices is increasingly crucial for joint slice analysis. Nevertheless, the tasks of learning joint embeddings and identifying shared and unique cell/domain types across ST slices remain challenging. To address this, we introduce a method called MaskGraphene, for the purpose of better aligning and integrating different ST slices using both self-supervised and contrastive learning. MaskGraphene learns the joint embeddings to capture the geometric information efficiently. MaskGraphene further facilitates spatial aware data integration and simultaneous identification of shared and unique cell/domain types across different slices. We have applied MaskGraphene to conduct integrative analyses on various types of ST datasets, including human cortex slices, mouse hypothalamus data, mouse sagittal brain sections, and mouse embryo developmental data. Across datasets, MaskGraphene successfully optimized joint embedding by introducing inter-slice connections, effectively performing batch correction and capturing shared tissue structures across different slices, and tracking spatiotemporal changes during mouse embryonic development.

Keywords: Spatial Transcriptomics · Batch correction · Data integration · Self-supervised learning · Contrastive learning.

1 Introduction

The intricate orchestration of biological processes within organisms hinges upon the diversity and specialization of individual cell types, each tailored to fulfill specific functions. In order to unravel the complexities of disease pathology and tissue functioning, understanding the connections and interactions of neighboring and distant cells is crucial since the behavior of these cells is profoundly influenced by their microenvironment.[1] While single-cell RNA sequencing (scRNA-seq) techniques have revolutionized our ability to characterize cells at an unparalleled per-cell resolution, the absence of spatial information has limited our exploration of using transcriptional profiles to acquire more accurate analysis.[2]

Advancements in Spatial Transcriptomics (ST) have bridged this gap by facilitating the simultaneous measurement of mRNA expression and spatial coordinates within tissue sections.[2] ST techniques have significantly accelerated our exploration of the complex transcriptional landscape within heterogeneous tissues.[3,4] These techniques fall into two main categories: fluorescence-based in situ transcriptomics, such as single-molecule fluorescent in situ hybridization (smFISH), STARmap, and MERFISH, and spatial barcoding coupled with next-generation sequencing, exemplified by methods like Slide-seq (per cell resolution), 10x Visium (1-10 cells resolution), and spatial transcriptomics (ST).[5–9] However, integrated analysis of ST datasets produced by different conditions or technologies remains challenging.

In contrast to the conventional approach of single-slice ST data analysis focuses on unraveling spatial domain distribution,[10] there is a growing recognition of the value in integrative and comparative analyses of ST datasets originating from diverse sources, encompassing various individual samples, biological conditions, technological platforms, and developmental stages. This broader perspective offers a more comprehensive understanding of spatial tissue structures as well as introduces more available information to perform analysis. However, similar to integrative analyses in scRNA-seq data, ST slices can exhibit serious batch effects that potentially obscure genuine biological signals, complicating data interpretation and integration. A proper multi-slice ST data integration analysis could naturally minimize the batch

effect by jointly considering and processing signals from each sample.

However, some existing integration strategies often prioritize aggregating gene expression data across slices or using batch correction techniques primarily designed for scRNA-seq, like Harmony[11] and Seurat.[12] These approaches frequently overlook the vital spatial coordinates. PRECAST[13] and DeepST[14] are recent methods that bring fresh insights into the integration of spatial transcriptomics data, aiming to overcome the challenges faced by earlier strategies. PRECAST employs a Gaussian Mixture Model and Hidden Markov Random Field to derive latent joint embeddings. DeepST leverages a Variational Graph Autoencoder and integrates data augmentation techniques for joint ST slice analysis. However, they often fail to generate good joint node embeddings and achieve precise batch correction and integration. STAligner, proposed recently, utilizes a Graph Attention Autoencoder combined with a Triplet loss strategy. Although STAligner has demonstrated its strength in multi-slice analyses to some extent, it shares a significant limitation with many graph-based deep learning models: an inability to learn joint node embeddings to capture precise geometric information for accurate node-to-node alignment across different slices, which limits its ability in ST data integration and batch correction. Another tool, PASTE[15] is able to align and integrate ST data from multiple adjacent tissue slices and achieves a nearly 1:1 node-to-node matching ratio, however, it is incapable of generating joint node embeddings for further batch correction analysis due to the formulation of its Optimal Transport problem. Moreover, PASTE is only applied to the integration scenario when different ST slices are adjacent within the tissue.

To deal with existing challenges among these methods, we introduce MaskGraphene, a graph neural network with both self-supervised and self-contrastive training strategies designed for aligning and integrating ST data with gene expression and spatial location information while generating batch-corrected joint node embeddings. Specifically, to link two slices and reach a good node-to-node matching ratio, MaskGraphene employs two different types of inter-slice connections in the graph model via "hard-links" and "soft-links" depending on the integration scenarios. We benchmarked MaskGraphene against several methods including PRECAST,[13] STAligner,[16] and DeepST[14] across a diverse range of ST datasets. These include human dorsolateral prefrontal cortex (DLPFC) slices from multiple samples,[17] mouse hypothalamus slices,[18] sagittal anterior and posterior slices of the mouse brain, and a developmental atlas of mouse organogenesis. MaskGraphene achieves the best node-to-node matching ratio among all benchmarked tools and its joint node embeddings in 2D UMAP visualizations can capture the whole slice shape, layer-wise pattern and shape, and spatial relations in an extraordinary manner. MaskGraphene also consistently exhibits exceptional performance in comprehensive spatial data integration and batch correction, simultaneous identification of unique and shared tissue structures across different slices, and downstream comparative analysis.

2 Method

2.1 MaskGraphene workflow

In the workflow (Figure 1), MaskGraphene takes spots-genes expression matrix and spatial coordinates from two tissue slices as the input. MaskGraphene first constructs k-NN networks between spots based on their spatial coordinates for both slices. Using a graph attention auto-encoder backbone[19,20] MaskGraphene then optimizes low-dimensional latent embeddings by minimizing the reconstruction loss with masked self-supervised learning strategy.[21–23] To enhance the inter-slice connections and improve joint embeddings, MaskGraphene employs different linking approaches to join two slices by either "hard" or "soft" links to handle two different ST integration scenarios. Using the joint embeddings, MaskGraphene can seamlessly integrate ST datasets, enabling the identification of similar spatial domains or cell types from different tissue slices. We demonstrated the capability of MaskGraphene in several downstream analyses including batch correction with layer-wise alignment accuracy and spot-to-spot (node-to-node) matching ratio, UMAP visualization for batch correction, spatial domain identification with joint embeddings across slices, stitching mouse brain anterior and posterior brain ST slices, and integrating ST slices across different mouse embryo development stages.

2.2 Graph input processing and two types of inter-slice connections

MaskGraphene processes gene expression data and spatial coordinates from two ST slices. First, it normalizes raw gene expressions based on library size and log-transforms them using the SCANPY package.[24] It then identifies the highly variable genes (HVGs) from both slices, and focuses on their HVGs intersection to maintain the consistency of expression features. Leveraging the spatial coordinates, MaskGraphene

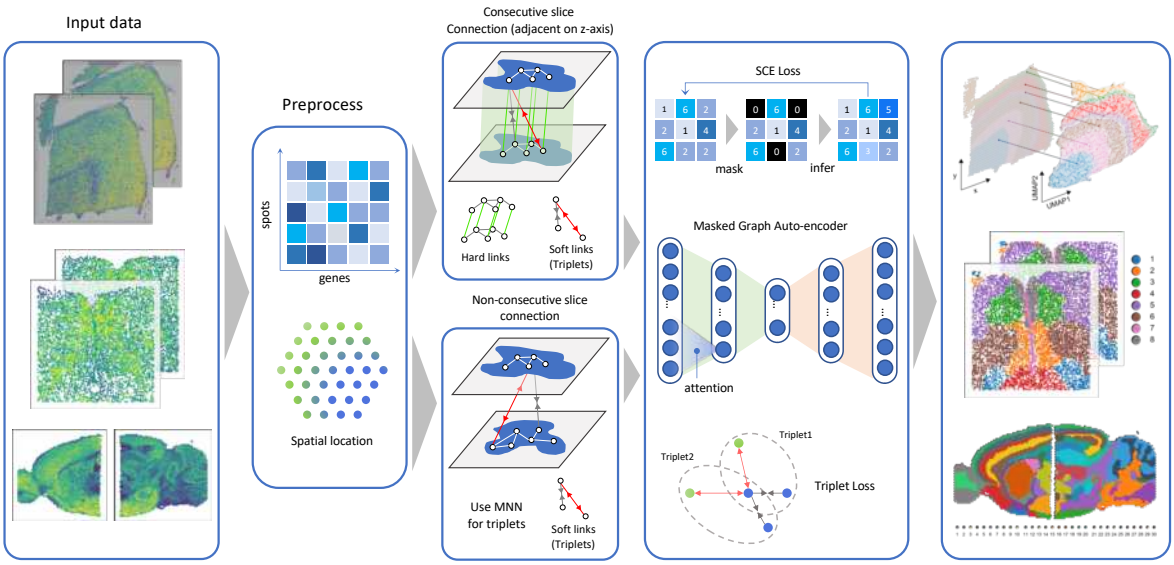

Figure 1. MaskGraphene workflow. MaskGraphene normalizes the expression profiles from all spots and creates a spatial network using their coordinates. Based on the priors of the slice relationship, we build "hard" and "soft" links or only "soft" links to enhance two different types of inter-slice connections. MaskGraphene then uses a graph neural network to create a spatially informed joint embedding by optimizing the reconstruction loss with masked gene expression features. For "hard" links, MaskGraphene integrates node-to-node matching links from an in-house ST local alignment algorithm. For "soft" links via triplet loss, MaskGraphene selects spots across slices as triplets based on their embeddings, with the goal of bringing similar spots closer and pushing different spots further apart in an iterative manner. This process continues until all spots are appropriately aligned and batch-corrected.

calculates spot similarities via Euclidean distance and constructs a k-NN graph with only intra-slice connections accordingly.

As mentioned previously, MaskGraphene enhances inter-slice connections based on two integration scenarios. When multiple consecutive ST slices (specifically for adjacent slices in the z-axis) from the tissue are made available, MaskGraphene will adopt "hard-links" into k-NN networks to form direct links between spots from different slices. These "hard-links" can be generated from existing tools that perform spot-to-spot alignment across slices like Paste.[15] In addition, to improve these "hard-links" from consecutive slices, we also designed an in-house spot-to-spot local alignment algorithm, to prepare inter-slice "hard-links" for MaskGraphene. The output of PASTE and our local alignment algorithm is an alignment mapping matrix for every two consecutive ST slices. MaskGraphene constructs spot-to-spot links across slices relying on the nonzero πs for each spot/cell from the alignment mapping matrix. π determines the mapping between spot i from the first slice and spot j from the second slice. To further enhance the connection between two consecutive slices, MaskGraphene will utilize mutual nearest neighbors (MNNs)[25] from two slices to construct spot triplets relying on their embeddings during training. We denote the inter-slice links by MNNs as "soft-links" since they are not directly integrated into the k-NN networks like "hard-links", but strengthened in the triplet loss, which is often used in single-cell RNA-seq batch correction.[16,26,27] These "soft-links" were designed to emphasize differences between dissimilar spots across slices while simultaneously clustering similar spots. Essentially, the goal is to bring alike spots nearer and set apart the contrasting ones.[26] Finally, to improve the data quality since the dropout rate of most ST data is high, MaskGraphene imputes the zero values of each spot depending on spot-to-spot links across slices.

When two slices are not consecutive (not adjacent in the z-axis) but batch correction and integration are favorable to integrate common domains or cell types across different sample slices (all the rest of integration application scenarios), MaskGraphene will only employ "soft-links" to strengthen the connections because the "hard-links" by our in-house local alignment algorithm or PASTE aims to generate a strong nearly 1:1 node-to-node mapping which does not exist in this integration scenario.

2.3 Graph model backbone

In this section, we present the mathematical formulation and details of our backbone, graph attention autoencoder. The graph attention autoencoder has three major components, which are the encoder, the decoder, and the graph attention mechanism.

Encoder. The encoder generates node (spot) embeddings by aggregating information from all neighbors. We denote $\mathbf{h}_u^{(0)}$ as the feature of spot u. The lth encoder layer generates the embedding of spot u in layer l as follows:

$$\mathbf{h}_u^{(l)} = \epsilon\Big(\sum_{v\in\mathcal{N}_u} \alpha_{uv}^{(l)}(\mathbf{W}^{(l)}\mathbf{h}_v^{(l-1)})\Big) \tag{1}$$

where $\alpha_{uv}^{(l)}$ is the attention coefficient, discussed further below, ϵ is the activation function, $\mathcal{N}_u$ denotes all the neighbors of node u including u itself, $\mathbf{h}_v^{(l-1)}$ denotes the embedding of node v in the $l-1$ layer, and $\mathbf{W}^{(l)}$ is the matrix of trainable parameters in l^{th} layer.

Graph Attention Mechanism. The attention coefficient $\alpha_{uv}^{(l)}$ is calculated by (2) and (3) where $\oplus$ is the concatenation operation and σ is a sigmoid activation function.

$$\alpha_{uv}^{(l)} = \frac{\exp\left(\sigma({\mathbf{a}^{(l)}}^T\mathbf{g}_{uv}^{(l)})\right)}{\sum_{v'\in\mathcal{N}_u}\exp\left(\sigma({\mathbf{a}^{(l)}}^T\mathbf{g}_{uv'}^{(l)})\right)} \tag{2}$$

$$\mathbf{g}_{uv}^{l} = \mathbf{W}^{(l)}\mathbf{h}_u^{(l-1)} \oplus \mathbf{W}^{(l)}\mathbf{h}_v^{(l-1)} \tag{3}$$

The attention mechanism here is exploited to strengthen the connection between nodes that are represented by similar expression profiles. The attention coefficient $\alpha_{uv}^{(l)}$ indicates the different contributions of each neighbor used in the aggregation process. The weights of edges are automatically calculated, based on node embedding.

Decoder. The decoder attempts to reconstruct the normalized expression profile for each spot u given the latent embeddings of the encoder. The lth layer of the decoder (from the perspective of spot u) defined as:

$$\hat{\mathbf{h}}_u^{(l)} = \epsilon\Big(\sum_{v\in\mathcal{N}_u} \hat{\alpha}_{uv}^{(l)}(\hat{\mathbf{W}}^{(l)}\hat{\mathbf{h}}_v^{(l-1)})\Big) \tag{4}$$

where $\hat{\alpha}_{uv}^{(l)}$ is the decoder attention coefficient which is calculated similarly as $\alpha_{uv}^{(k)}$ in the encoder, and $\hat{\mathbf{W}}^{(l)}$ is the matrix of trainable parameters in l^{th} layer.

The reconstruction loss function will be introduced in the next section after introducing masked self-supervised loss.

2.4 Masked self-supervised loss

For our graph model backbone, we introduced a self-supervised loss to reconstruct the node features that are randomly masked from the input gene expression matrix. Specifically, we first perturb the input graph by masking node features and then attempt to reconstruct the original input. This strategy aims to enforce the reconstruction in the embedding space and regularize the node feature reconstruction.

For each node i, let $\mathbf{z}_i$ represent its reconstructed gene expression via the decoder. The reconstruction loss function is formulated as:

$$\mathcal{L}_{\text{masked}} = \frac{1}{|\bar{V}|}\sum_{v_i\in\bar{V}}\left(1-\left(\frac{{\mathbf{x_i}}^T\mathbf{z_i}}{||\mathbf{x_i}||\cdot||\mathbf{z_i}||}\right)\right)^{\gamma}, \gamma\geq 1, \tag{5}$$

where $\mathbf{x}_i$ denotes the original normalized expression profile for each spot, $\bar{V}$ is the subset of spots with masked features in the graph, and γ is a scaling factor for adjusting loss weight. Through this scaled cosine error loss to measure the reconstruction error, the model strives to learn node embeddings under a masked autoencoder.

To further utilize the feature information, there is a supportive network that serves as the target generator to produce latent prediction targets from the unmasked graph. This generator has an identical structure to the aforementioned encoder and projector but with a different set of trainable parameters, and the projector is a multi-layer perceptron (MLP). We learn the parameters of the encoder and projector by minimizing the following loss function:

$$\mathcal{L}_{\text{latent}} = \frac{1}{|N|}\sum_{i}^{N}\left(1-\left(\frac{\bar{\mathbf{z}}_\mathbf{i}^T\bar{\mathbf{x}}_\mathbf{i}}{||\bar{\mathbf{x}}_\mathbf{i}||\cdot||\bar{\mathbf{z}}_\mathbf{i}||}\right)\right)^{\gamma}, \gamma\geq 1, \tag{6}$$

where N denotes all the spots in the unmasked graph, $\bar{\mathbf{z}}_i$ represents the projected latent targets from the masked graph, and $\bar{\mathbf{x}}_i$ denotes the projected latent targets from the unmasked graph.

We are simultaneously optimizing two objective functions with a balancing coefficient λ:

$$\mathcal{L}_{\text{MaskGraphene}} = \mathcal{L}_{\text{masked}} + \lambda\mathcal{L}_{\text{latent}} \tag{7}$$

One intuition behind this is that we are enforcing the network not to smooth out those singular yet critical features from the gene expression profiles while suppressing the noises from batch effect, etc.

2.5 Triplet loss

To enforce the inter-slice connections by "soft-links", we introduce triplet loss, a fundamental component

of our proposed framework to learn joint embeddings when two different slices are not consecutive.

Triplet Loss is based on the idea of triplets, which consist of an anchor spot $\mathbf{a}$, a positive spot $\mathbf{p}$, and a negative spot $\mathbf{n}$. For each triplet $(\mathbf{a}, \mathbf{p}, \mathbf{n})$, the loss is defined as follows:

$$\mathcal{L}_{\text{triplet}} = \max\left(\mathit{Distance}(\mathbf{a}, \mathbf{p}) - \mathit{Distance}(\mathbf{a}, \mathbf{n}) + \alpha, 0\right) \quad (8)$$

where distance$(\mathbf{a}, \mathbf{p})$ is the Euclidean distance between the anchor spot $\mathbf{a}$ and the positive spot $\mathbf{p}$, distance$(\mathbf{a}, \mathbf{n})$ is the distance between the anchor spot $\mathbf{a}$ and the negative spot $\mathbf{n}$, α is a margin that controls the minimum difference required between the distances of similar and dissimilar pairs. To construct triplets, we first determine mutual nearest neighbors (MNNs) across two slices. MNNs are constructed by taking the pairs of spots from distinct slices that are mutually k-nearest-neighbors.[28] After we identify MNNs matches between two slices, we define each MNN match as the $(\mathbf{a}, \mathbf{p})$ pair. We further take each spot from MNN matches of one slice as an anchor spot and randomly select any spot from the other slice as a negative spot to form the $(\mathbf{a}, \mathbf{n})$ pair. Triplet Loss encourages the model to pull the anchor and positive spots (MNNs across slices) closer in the embedding space while pushing the anchor and negative spots (dissimilar spots across slices) farther apart.

3 Results

3.1 Benchmark Datasets

In this study, we used dorsal lateral prefrontal cortex (DLPFC) 10x Visium datasets, mouse hypothalamus MERFISH datasets, mouse brain sagittal sections 10x Visium datasets, and mouse embryo development Stereo-seq datasets. Detailed information is listed in Table S1. The DLPFC dataset includes 12 human DLPFC sections with manual annotation (from cortical layers 1 to 6 and white matter (WM)), taken from three individual samples,[17] and each sample contains 4 consecutive slices. The mouse hypothalamus datasets contain five manually annotated consecutive slices.[18] The two mouse embryo Stereo-seq datasets were acquired at two different time points at E11.5 and E12.5. These datasets are from a large stereo-seq project called MOSTA:[29] Mouse Organogenesis Spatiotemporal Transcriptomic Atlas by BGI. Lastly, the mouse brain sagittal sections are divided into posterior and anterior slices.

We benchmarked MaskGraphene against three other state-of-the-art methods, STAligner, PRECAST, and DeepST, which are capable of processing joint ST slice analysis and performing batch correction.

3.2 Experimental Setup for MaskGraphene

For MaskGraphene, we used the Adam optimizer[30] to minimize the loss with an initial learning rate of $1e^{-3}$ and a weight decay of $1e^{-5}$. The default number of iterations was set to 2000 for training with masked self-supervised loss and 500 for training with triplet loss. We used a 2-layer graph attention networks (GATs) for both the encoder and the decoder in MaskGraphene's backbone. The mask rate and remask rate for input features of first layer of encoder and decoder were set to 0.5 and 0.1. The dimension of the encoded hidden features, used for clustering spots, was set to 32. The input dimension of the hidden layer was set to 512. Since this tool has a general autoencoder structure, the input feature size of the encoder layers was equal to the output feature size of the corresponding decoder layers, and vice versa.

3.3 MaskGraphene enhances joint embedding to capture geometric information

To perform joint consecutive slice analysis, we first used nine pairs of DLPFC slices. As introduced in the Methods section, MaskGraphene integrates "hard-links" to directly enhance the inter-slice connections to learn joint spot embeddings. To investigate whether MaskGraphene generates better joint spot (node) embeddings and improves the integration of consecutive slices compared to three other tools, we performed two evaluation experiments. The first essential evaluation analysis for layer-wise alignment accuracy we used relies on an important hypothesis that aligned spots from consecutive slices are more likely to belong to the same spatial domain or cell type. We utilized joint embeddings learned from all tools to align (anchor) spots from the first slice to (aligned) spots on the second slice for each slice pair. In Figure 2a-i, we compared the layer-wise alignment accuracy of four methods on all nine DLPFC slice pairs. Since DLPFC data has a unique layered structure, we meticulously designed this evaluation metric to demonstrate whether anchor and aligned spots belong to the same layer (layer shifting = 0) or they belong to different layers (layer shifting = 1 to 6). We expected a good integration tool would demonstrate high accuracy for anchor and aligned spots belonging to the same layer (layer shifting = 0). In all nine DLPFC slice pairs, MaskGraphene achieved the best layer-wise alignment accuracy. For all tools, the majority of the anchor and aligned spots across two consecutive slices either originated from the same layer or two neighbor layers.

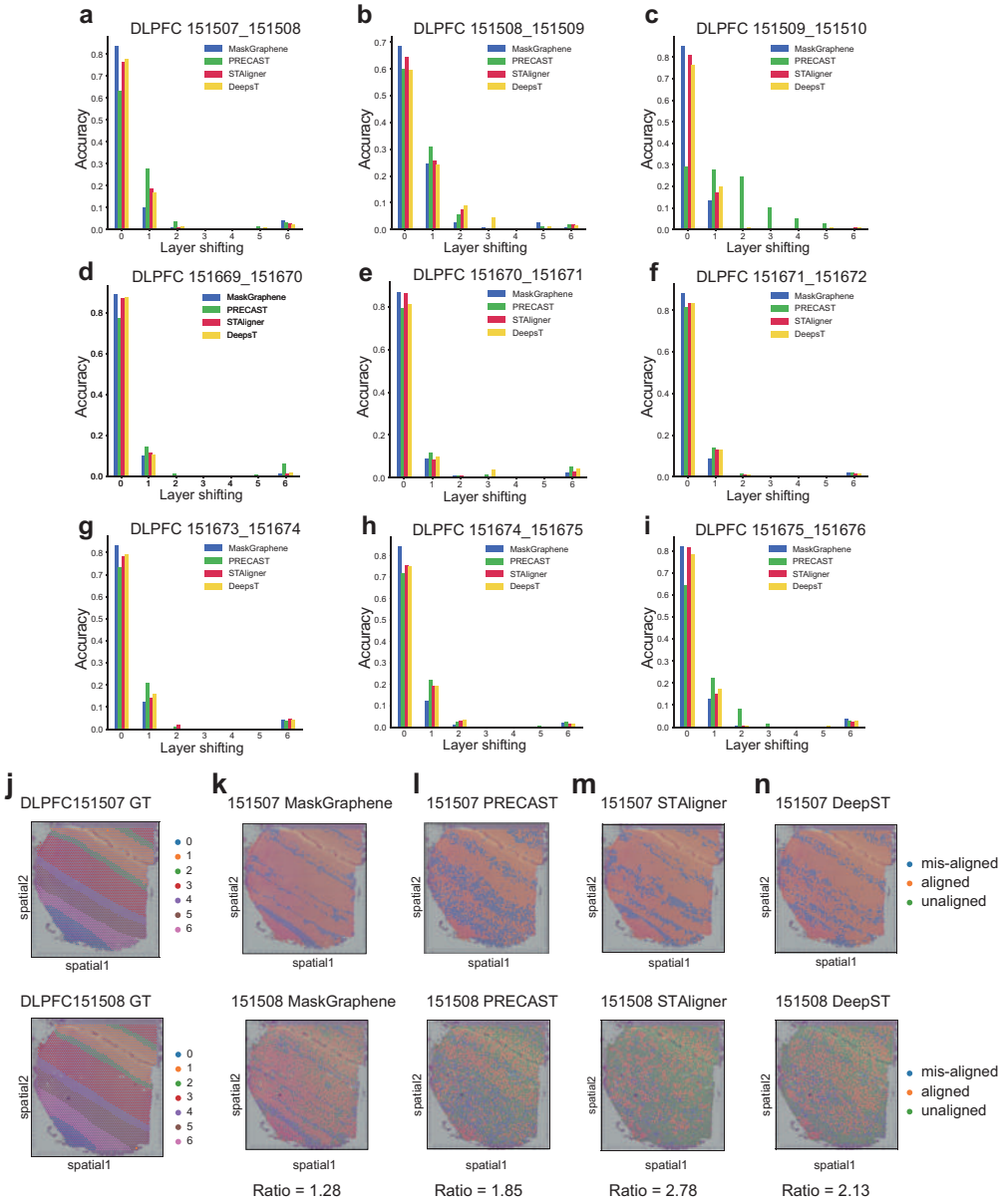

Figure 2. Bar plots for layer-wise alignment accuracy and visualization plots for alignment-misalignment-unalignment. (a-i) Bar plots depict the layer-wise alignment accuracy of four methods on DLPFC data based on seven different layer-shifting conditions. **(j)** Manual annotation of DLPFC layers on sections 151507 and 151508. **(k-n)** These visualization plots show aligned spots, misaligned spots, and unaligned spots when performing spot-to-spot matching from DLPFC 151507 to 151508 based on the joint node embeddings of four methods. Values below each plot represent the spot-to-spot matching ratio.

Although the layer-wise alignment accuracy quantifies the spot-to-layer alignment accuracy to some extent, it is crucial to assess the spot-to-spot (node-to-node) alignment accuracy based on joint spot embeddings. To further investigate this, we plotted anchor and aligned spots on both slices and divided them into three categories: aligned spots, misaligned spots, and unaligned spots. We still used the layer labels to define aligned and misaligned spots. As illustrated in Figure 2j-n, all four methods demonstrated a high proportion of unaligned spots on the second slice for the DLPFC 151507 and 151508 pair, which indicated that all tools had a bias to align multiple anchor spots from the first slice to the same spot of the second slice, and left a substantial proportion of spots unaligned on the second slice. Aligned

spots from both slices originate from the same layer. We calculated the node-to-node mapping ratio and observed that MaskGraphene had the lowest ratio (1.28), followed by PRECAST (1.85), DeepST (2.13), and STAligner (2.78). On average, 1.28 spots from the first slice were aligned to the same spot of the second slices in MaskGraphene, however, 2.78 spots from the first slice were aligned to the same spot from the second slices in STAligner. For deep learning-based methods, it is common for spots in low-dimensional space to lose some geometric information from the original gene expression and spatial coordinate profile during the optimization process. These tools are thus more likely to maintain a worse spot-to-spot alignment performance, but a better layer-wise alignment accuracy. However, by introducing "hard-links" to strengthen inter-slice connections, we imposed additional constraints on the optimization process, leading to better joint embeddings to capture the original geometric information for spot-to-spot alignment in MaskGraphene. We further conducted this evaluation experiments on all rest eight pairs of DLPFC slices (Figure S1-S5). For the overall performance, MaskGraphene reached a ratio ranging from 1.28 to 1.46, PRECAST reached a ratio ranging from 1.82 to 1.93, STAligner reached a ratio ranging from 2.64 to 3.76, and DeepST reached a ratio ranging from 2.04 to 2.48. MaskGraphene achieved the smallest ratio in all nine pairs of DLPFC slices.

Furthermore, across all nine pairs, we observed that the misaligned spots (in blue color in Figure 2j-n and Figure S1-S5) on the first slice were only clearly aggregated along the layer boundaries in MaskGraphene, however, these misaligned spots were also scattered within the layers in PRECAST, STAligner, and DeepST. Even though PRECAST had the second smallest ratio for five pairs, it displayed the worst scattered pattern for misaligned spots. These scattered misaligned spots indicated a bad batch correction effect in the other three tools, especially in PRECAST.

3.4 MaskGraphene improves the batch correction and integration of consecutive slices

Once joint node embeddings are generated, we can further evaluate the batch correction from the integration of two consecutive slices. We visualized the joint embeddings for batch correction between cell/domain types using UMAP for each method. For the DLPFC 151507 and 151508 pair (Figure 3a), the UMAP plots for MaskGraphene showed that spots from two different slices were evenly mixed (Figure 3a, right panel) while the predicted domain clusters were well segregated, highly concordant with the ground truth (Figure 3, middle panel versus left panel), and there were significant improvements in visualization in comparison to the three other tools. Inspecting the UMAP visualizations further, it was intriguing to see that the UMAP plot by MaskGraphene could recover the whole slice shape, layer-wise pattern and shape, and spatial relations in DLPFC sections 151507 and 151508 (see the ground truth patterns in Figure 2j). We further demonstrated this UMAP analysis for the DLPFC 151675 and 151676 pair and all the rest of the pairs, and plotted layer-wise annotations by ground truth and MaskGraphene (Figure 3c-d and Figure S6-S7). We observed that the UMAP plots by MaskGraphene recovered the whole slice shape, layer-wise pattern and shape, and spatial relations very well in all DLPFC pairs. These UMAP results further proved that the joint embeddings by MaskGraphene captured the original geometric information in an extraordinary manner. To our knowledge, no existing tools can achieve such superior joint embeddings for UMAP visualization in either multi-slice or single-slice DLPFC analyses.

For four pairs of mouse hypothalamus slices, we observed a similar node-to-node mapping ratio in all four tools (1.92 in PRECAST, 2.08 in DeepST, 2.14 in MaskGraphene, and 2.53 in STAligner). When we investigated misaligned spots on all pairs, we observed a similar pattern as in DLPFC: the misaligned spots on the first slice were clearly aggregated along the layer boundaries in MaskGraphene, however, these misaligned spots were scattered within the cluster in PRECAST, STAligner, and DeepST. These results indicated a bad batch correction in the other three tools since scattered misaligned spots/cells could not be differentiated within clusters. Taking the mouse hypothalamus pair -0.19 and -0.24 as an example (Figure 3b), MaskGraphene did exhibit an exceptional batch correction in comparison to the other three tools. The joint embeddings by MaskGraphene still captured the original geometric information to some extent, although this effect is inferior compared to the DLPFC data. By investigating the visualization plots by ground truth, PRECAST, STAligner, and DeepST all exhibited mixed clusters annotated by the ground truth that can not be differentiated by them. As we expected based on misaligned spots, PRECAST showed the worst batch correction since spots from different cell/domain types were misaligned (mixed) when labeled by ground truth.

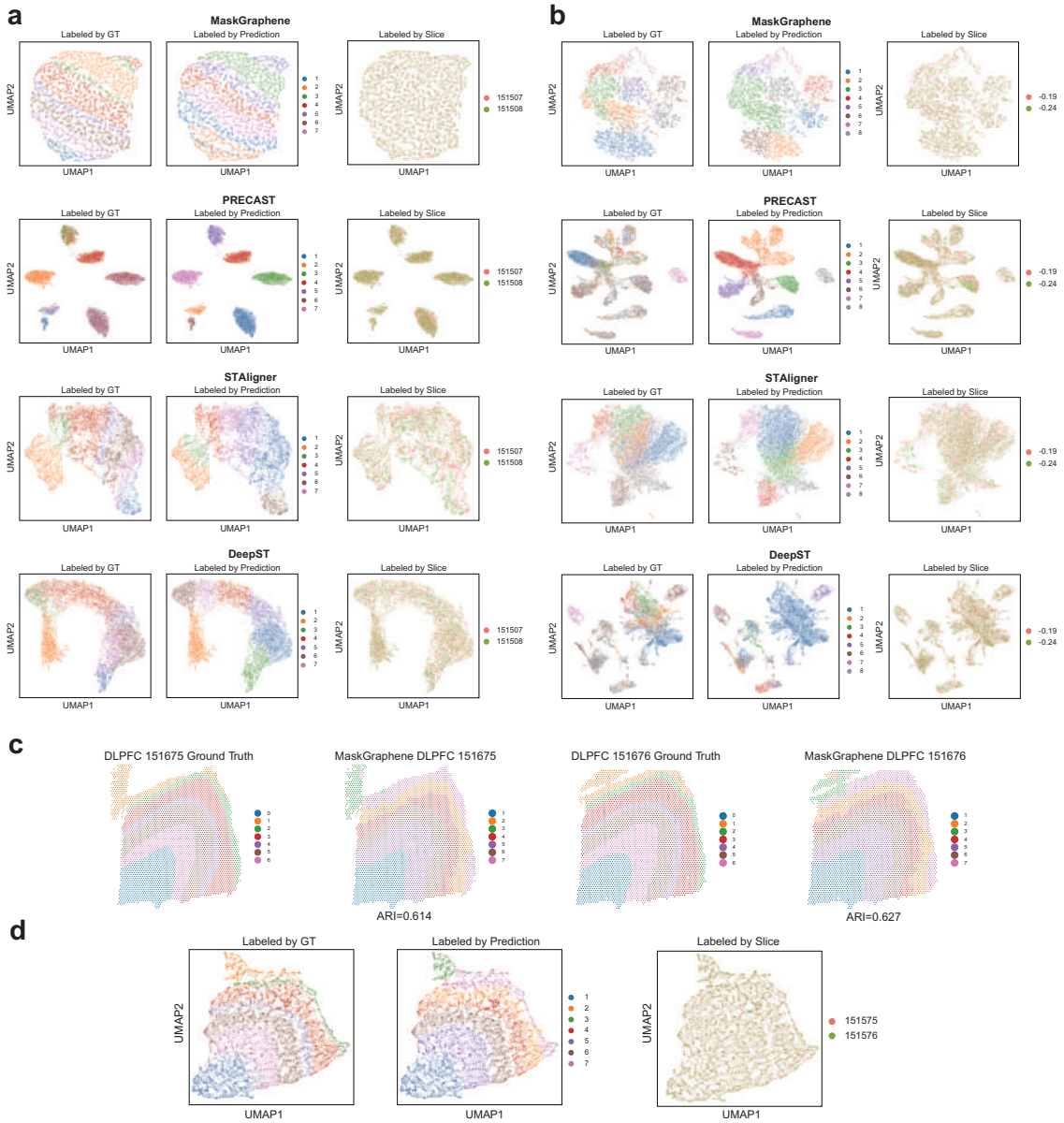

Figure 3. UMAP plots of low dimensional joint embedding distribution for batch correction. These UMAP plots depict the 2D distribution of latent joint embeddings by four methods on **(a)** the DLPFC pair 151507 and 151508, and **(b)** the mouse hypothalamus pair -0.19 and -0.24. Each subfigure contains colored spots labeled by three different setups: ground truth (GT), method prediction, and slice index. **(c)** The manual annotation and MaskGraphene's prediction labels for DLPFC layers on sections 151675 and 151676. **(d)** The UMAP plots for latent joint embeddings by MaskGraphene on the DLPFC pair 151507 and 151508.

3.5 MaskGraphene enhances domain identification through joint embedding optimization

Integrating data from multiple ST slices can allow us to estimate joint embeddings of expressions representing variations between cell or domain types across slices, which has the potential to provide us a way to better detect spatial domains or cell types compared to single slice analysis.[15,16] To further quantitatively compare the effectiveness of these methods in capturing spatial domains via joint embeddings, we employed joint embeddings from two slices in the mouse hypothalamus dataset to perform clustering together using the clustering method mclust.[31] We then employed the Adjusted Rand Index (ARI) as the evaluation metric to compare the clustering

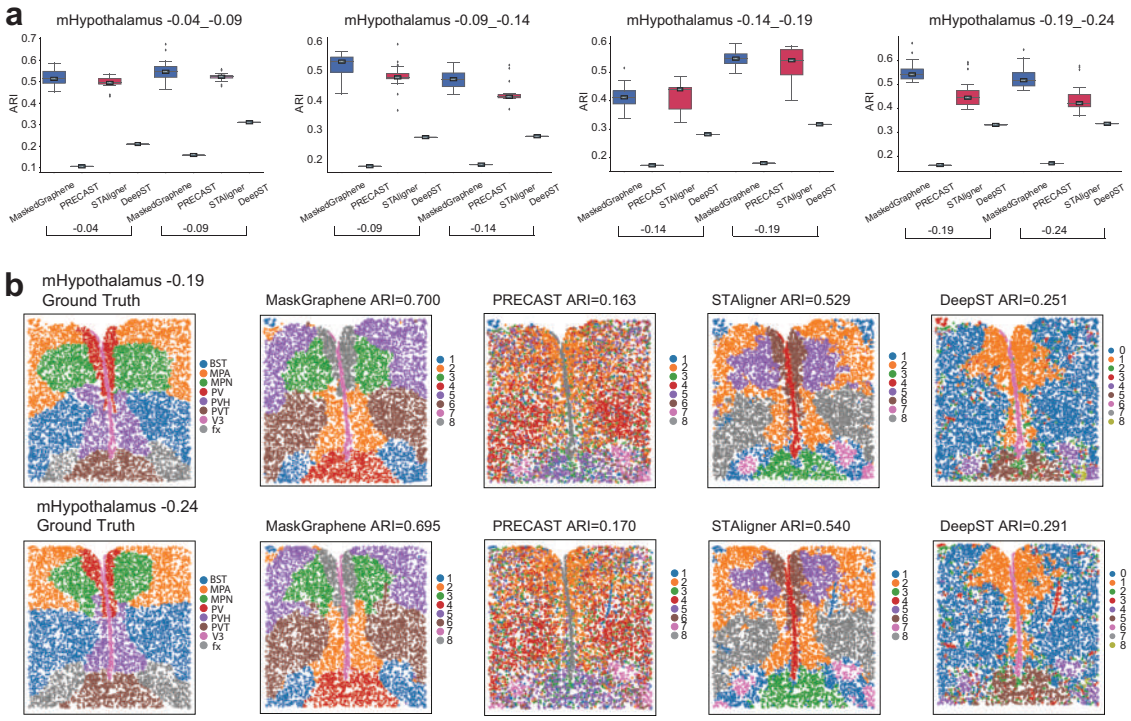

Figure 4. ARI boxplots and visualization plots for domain identification for mouse hypothalamus dataset. (a) ARI boxplots for four mouse hypothalamus pairs by four methods. The boxplots are based on 20 runs for each method. **(b)** Visualization plots for eight clusters by both ground truth and the predictions from four methods.

results of each tool with the ground truth in each slice, with higher ARI scores indicating better domain identification. In Figure 4a, the ARI boxplots indicated that MaskGraphene achieved the best ARI on the mouse hypothalamus pairs -0.04 and -0.09 (average ARI=0.513, 0.546), -0.09 and -0.14 (average ARI=0.532, 0.472), and -0.19 and -0.24 (average ARI=0.542, 0.518), and second best ARI on the -0.14 and -0.19 pair (average ARI=0.412, 0.548 in MaskGraphene; average ARI=0.440, 0.542 in STAligner). By investigating the spatial domains identified by each tool for the mouse hypothalamus pair -0.19 and -0.24, we observed that only MaskGraphene identified all eight spatial domains in the ground truth with clear boundaries for both slices (Figure 4b).

3.6 MaskGraphene stitches mouse brain anterior and posterior sections

So far, we have investigated the capability of MaskGraphene in integrating consecutive sample slices. In this section, we further revealed its power in integrating non-consecutive slices via "soft-links" and employed two 10x Visium datasets of mouse brain sagittal sections, divided into posterior and anterior portions. To evaluate the batch correction accuracy of all methods, we utilized the Allen Brain Atlas as a reference (Figure 5a) and visually compared the clustering results of MaskGraphene against the other three methods (Figure 5b-e). PRECAST exhibited the worst performance, generating domains with scattered noises. Specifically, PRECAST was not able to detect and connect common spatial domains along the shared boundary of the anterior and posterior sections. MaskGraphene, STAligner, and DeepST demonstrated their capabilities in detecting and connecting common spatial domains along the shared boundary. Particularly noteworthy was their performance in the cerebral cortex (CTX) region, where both MaskGraphene and STAligner outperformed DeepST and PRECAST by accurately identifying and aligning six distinct layers across the anterior and posterior sections. Moreover, for unshared regions, both MaskGraphene and STAligner excelled in separating the caudal putamen (CP) and nucleus accumbens (ACB) and distinguishing the layers within the cerebellar cortex (CBX). MaskGraphene also identified a coherent arc across two sections for CA1, CA2, and CA3. These findings underscore the performance of MaskGraphene in capturing spatial

structures within complex datasets and its competence in batch correction for non-consecutive slices.

3.7 MaskGraphene aligns tissues and organs across different developmental stages

Lastly, we demonstrated MaskGraphene's ability to integrate two slices from different development stages for joint analysis, to study the spatiotemporal development in tissue structures during mouse organogenesis. In Figure 5h, the two mouse embryo slices were acquired at two different time points (E11.5 and E12.5), and MaskGraphene identified a total of 15 clusters. Despite the differences in the sizes of the two slices and the presence of noticeable batch effects, MaskGraphene effectively harmonized the data by integrating them into a common embedding space and detected both shared (labeled with the same color across slices by MaskGraphene) and developing structures across different time points. In Figure 5g, we highlighted five region-based annotations such as dorsal root ganglion, brain, heart, jaw and teeth, and liver for both slices. MaskGraphene successfully retrieved these five shared structures in both slices. We also observed that at developmental stage E11.5, structures like the ovary and pancreas were less developed compared to E12.5. These results facilitated the reconstruction of the developmental progression of each tissue structure throughout organogenesis.

4 Discussion and Conclusion

In this work, we developed, MaskGraphene, a graph neural network with both self-supervised and self-contrastive training strategies to align and integrate ST data using gene expression and spatial location information. To strengthen the interconnection of two slices, MaskGraphene employs "soft-links" and "hard-links" to either directly connect the k-NN graph of the model or indirectly connect the two graph network by triplet loss. MaskGraphene achieves batch-corrected joint node embeddings by preserving geometric information. We benchmarked MaskGraphene with other competing methods such as PRECAST, STAligner, and DeepST on several datasets, and designed different evaluation metrics to investigate the nature of joint node embeddings. MaskGraphene reached a more accurate node-to-node alignment by a lower node-to-node mapping ratio compared to the other three methods. Moreover, MaskGraphene showed favorable performance in terms of batch correction and integration when compared to the other three methods across different datasets.

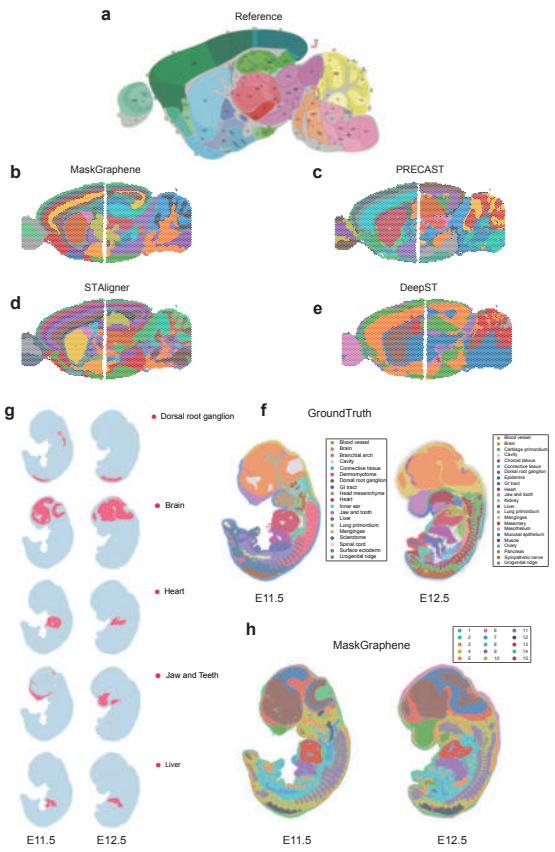

Figure 5. Visualization plots for batch correction and integration in mouse brain sagittal dataset and embryo dataset. (a) The Allen brain atlas annotation for the mouse brain sagittal section. **(b-e)** Domain identification by four methods in mouse brain sagittal dataset. **(f-g)** Domain identification by the ground truth in the mouse embryo dataset. **(h)** Domain identification by MaskGraphene in the mouse embryo dataset.

5 Acknowledgements

This work was supported by the NIGMS Maximizing Investigators' Research Award (MIRA) R35 GM146960.

6 Data and code availability

All original code has been deposited on our GitHub repository (https://github.com/maiziezhoulab/MaskGraphene) and is publicly available.

References

1. Perrimon, N., Pitsouli, C. and Shilo, B.-Z. (2012). Signaling mechanisms controlling cell fate and embryonic patterning. Cold Spring Harbor perspectives in biology, 4(8):a005975.
2. Marx, V. (2021). Method of the Year: spatially resolved transcriptomics. Nature Methods, 18(1):9–14.
3. Hu, Y., Zhao, Y., Schunk, C. T., Ma, Y., Derr, T. and Zhou, X. M. (2023). ADEPT: Autoencoder with differentially expressed genes and imputation for robust spatial transcriptomics clustering. Iscience, 26(6).
4. Dong, K. and Zhang, S. (2022). Deciphering spatial domains from spatially resolved transcriptomics with an adaptive graph attention auto-encoder. Nature communications, 13(1):1–12.
5. Chen, J., McSwiggen, D. and Ünal, E. (2018). Single molecule fluorescence in situ hybridization (smFISH) analysis in budding yeast vegetative growth and meiosis. JoVE (Journal of Visualized Experiments), None (135):e57774.
6. Wang, X., Allen, W. E., Wright, M. A., Sylwestrak, E. L., Samusik, N., Vesuna, S., Evans, K., Liu, C., Ramakrishnan, C., Liu, J. et al. (2018). Three-dimensional intact-tissue sequencing of single-cell transcriptional states. Science, 361(6400):eaat5691.
7. Moffitt, J. R. et al. (2016). High-throughput single-cell gene-expression profiling with multiplexed error-robust fluorescence in situ hybridization. Proceedings of the National Academy of Sciences, 113(39):11046–11051.
8. Rodriques, S. G. et al. (2019). Slide-seq: A scalable technology for measuring genome-wide expression at high spatial resolution. Science, 363(6434):1463–1467.
9. Ståhl, P. L., Salmén, F., Vickovic, S., Lundmark, A., Navarro, J. F., Magnusson, J., Giacomello, S., Asp, M., Westholm, J. O., Huss, M. et al. (2016). Visualization and analysis of gene expression in tissue sections by spatial transcriptomics. Science, 353(6294):78–82.
10. Zhou, X. (2022). Graphing cell relations in spatial transcriptomics. Nature Computational Science, 2(6): 354–355.
11. Korsunsky, I., Millard, N., Fan, J., Slowikowski, K., Zhang, F., Wei, K., Baglaenko, Y., Brenner, M., Loh, P.-r. and Raychaudhuri, S. (2019). Fast, sensitive and accurate integration of single-cell data with Harmony. Nature methods, 16(12):1289–1296.
12. Stuart, T., Butler, A., Hoffman, P., Hafemeister, C., Papalexi, E., Mauck, W. M., Hao, Y., Stoeckius, M., Smibert, P. and Satija, R. (2019). Comprehensive integration of single-cell data. Cell, 177(7):1888–1902.
13. Liu, W., Liao, X., Luo, Z., Yang, Y., Lau, M. C., Jiao, Y., Shi, X., Zhai, W., Ji, H., Yeong, J. et al. (2023). Probabilistic embedding, clustering, and alignment for integrating spatial transcriptomics data with PRECAST. Nature communications, 14(1):296.
14. Xu, C., Jin, X., Wei, S., Wang, P., Luo, M., Xu, Z., Yang, W., Cai, Y., Xiao, L., Lin, X. et al. (2022). DeepST: identifying spatial domains in spatial transcriptomics by deep learning. Nucleic Acids Research, 50(22):e131–e131.
15. Zeira, R., Land, M., Strzalkowski, A. and Raphael, B. J. (2022). Alignment and integration of spatial transcriptomics data. Nature Methods, 19(5):567–575.
16. Zhou, X., Dong, K. and Zhang, S. (2023). Integrating spatial transcriptomics data across different conditions, technologies and developmental stages. Nature Computational Science, 1–13.
17. Maynard, K. R., Collado-Torres, L., Weber, L. M., Uytingco, C., Barry, B. K., Williams, S. R., Catallini, J. L., Tran, M. N., Besich, Z., Tippani, M. et al. (2021). Transcriptome-scale spatial gene expression in the human dorsolateral prefrontal cortex. Nature neuroscience, 24(3):425–436.
18. Li, Z. and Zhou, X. (2022). BASS: multi-scale and multi-sample analysis enables accurate cell type clustering and spatial domain detection in spatial transcriptomic studies. Genome biology, 23(1):168.
19. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P. and Bengio, Y. (2017). Graph attention networks. arXiv preprint arXiv:1710.10903,.
20. Salehi, A. and Davulcu, H. (2019). Graph attention auto-encoders. arXiv preprint arXiv:1905.10715,.
21. Hou, Z., Liu, X., Cen, Y., Dong, Y., Yang, H., Wang, C. and Tang, J. (2022). Graphmae: Self-supervised masked graph autoencoders. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining 594–604, None.
22. Hou, Z., He, Y., Cen, Y., Liu, X., Dong, Y., Kharlamov, E. and Tang, J. (2023). GraphMAE2: A Decoding-Enhanced Masked Self-Supervised Graph Learner. In Proceedings of the ACM Web Conference 2023 737–746, None.
23. He, K., Chen, X., Xie, S., Li, Y., Dollár, P. and Girshick, R. (2022). Masked autoencoders are scalable vision learners. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition 16000–16009, None.
24. Wolf, F. A., Angerer, P. and Theis, F. J. (2018). SCANPY: large-scale single-cell gene expression data analysis. Genome biology, 19(1):1–5.
25. Haghverdi, L., Lun, A. T., Morgan, M. D. and Marioni, J. C. (2018). Batch effects in single-cell RNA-sequencing data are corrected by matching mutual nearest neighbors. Nature biotechnology, 36(5):421–427.
26. Yu, X., Xu, X., Zhang, J. and Li, X. (2023). Batch alignment of single-cell transcriptomics data using deep metric learning. Nature Communications, 14(1):960.
27. Deng, W., Zheng, L., Sun, Y. and Jiao, J. (2020). Rethinking triplet loss for domain adaptation. IEEE Transactions on Circuits and Systems for Video Technology, 31(1):29–37.
28. Hie, B., Bryson, B. and Berger, B. (2019). Efficient integration of heterogeneous single-cell transcriptomes using Scanorama. Nature biotechnology, 37(6):685–691.
29. Chen, A., Liao, S., Cheng, M., Ma, K., Wu, L., Lai, Y., Qiu, X., Yang, J., Xu, J., Hao, S. et al. (2022). Spatiotemporal transcriptomic atlas of mouse organogenesis using DNA nanoball-patterned arrays. Cell, 185(10): 1777–1792.

30. Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980,.
31. Fraley, C., Raftery, A. E., Murphy, T. B. and Scrucca, L. (2012). mclust version 4 for R: normal mixture modeling for model-based clustering, classification, and density estimation. Technical report Technical report.