DANCE: A Deep Learning Library and Benchmark for Single-Cell Analysis

Jiayuan Ding*[1](✉), Hongzhi Wen*[1], Wenzhuo Tang*[3], Renming Liu*[2], Zhaoheng Li[4], Julian Venegas[2], Runze Su[2,3], Dylan Molho[2], Wei Jin[1], Wangyang Zuo[5], Yixin Wang[6], Robert Yang[7], Yuying Xie[2,3](✉), and Jiliang Tang[1](✉)

[1] Department of Computer Science and Engineering, Michigan State University, USA
dingjia5@msu.edu, tangjili@msu.edu
[2] Department of Computational Mathematics, Science and Engineering, Michigan State University, USA
[3] Department of Statistics and Probability, Michigan State University, USA
xyy@msu.edu
[4] Biostatistics School Of Public Health, University of Washington, USA
[5] Department of Computer Science, Zhejiang University of Technology, China
[6] Department of Bioengineering, Stanford University, USA
[7] Johnson & Johnson, USA

Abstract. In the realm of single-cell analysis, computational approaches have brought an increasing number of fantastic prospects for innovation and invention. Meanwhile, it also presents enormous hurdles to reproducing the results of these models due to their diversity and complexity. In addition, the lack of gold-standard benchmark datasets, metrics, and implementations prevents systematic evaluations and fair comparisons of available methods. Thus, we introduce the DANCE platform, the first standard, generic, and extensible benchmark platform for accessing and evaluating computational methods across the spectrum of benchmark datasets for numerous single-cell analysis tasks. Currently, DANCE supports 3 modules and 8 popular tasks with 32 state-of-art methods on 21 benchmark datasets. People can easily reproduce the results of supported algorithms across major benchmark datasets via minimal efforts (e.g., only one command line). In addition, DANCE provides an ecosystem of deep learning architectures and tools for researchers to develop their own models conveniently. The goal of DANCE is to accelerate the development of deep learning models with complete validation and facilitate the overall advancement of single-cell analysis research. DANCE is an open-source python package that welcomes all kinds of contributions. All resources are integrated and available at https://omicsml.ai/.

Keywords: Single-cell Analysis · Deep Learning · Benchmarking.

* indicates equal contributions.

1 Introduction

Single-cell profiling technology has undergone rapid development in recent years, spanning from single modality profiling (RNA, protein, and open chromatin) [19,41,62,67,74,95,102,117,129], multimodal profiling [14,24,54,91,143] to spatial transcriptomics [7,22,28,89,96,130,138,142]. The fast revolution in this field has encouraged an explosion in the number of computational methods, especially machine learning-based methods. However, the diversity and complexity of current methods make it difficult for researchers to reproduce the results as shown in the original papers. The major challenges include no publicly available codebase, hyperparameter tuning, and differences between programming languages. Furthermore, a systematic benchmarking procedure is necessary to comprehensively evaluate methods since the majority of existing works have only reported their performance on limited datasets and comparison with insufficient methods. Therefore, a generic and extensible benchmark platform with comprehensive benchmark datasets and metric evaluation is highly desired to easily reproduce any algorithm other than state-of-art methods under different tasks across popular benchmark datasets via minimal efforts (e.g., only one command line). Considering deep learning methods like Graph Neural Networks [26,118,121,140,143] have shown promising performance in single-cell analysis, but the customized interfaces of such tools are largely missing in the existing packages. Those motivate the development of our DANCE system that not only acts as a benchmark platform, but also provides customized deep learning infrastructure interfaces to help researchers conveniently develop their models..

In this work, we present DANCE as a deep learning library and benchmark to facilitate research and development for single cell analysis. DANCE provides an end-to-end toolkit to facilitate single cell analysis algorithm development and fair performance comparison on different benchmark datasets. DANCE currently supports 3 modules, 8 tasks, 32 models and 21 datasets. Table 1 summarizes the key differences between DANCE and existing single-cell libraries and toolkits. The highlights of DANCE are summarized as follows:

- **Comprehensive Module Coverage:** Squidpy [101] proposes an efficient and scalable infrastructure only for spatial omics analysis. DeepCell [136] forms a deep learning library for single-cell analysis but only biological images are covered. The library specializes in models for cell segmentation and cell tracking. Even though the popular Scanpy [144] provides a powerful tool for single-cell analysis spanning all modules, it focuses on the field of data preprocessing instead of modeling. Similarly, even though Seurat [54] touches on all three modules, its R language-based interface restricts its applicability for the development of deep learning methods due to limited R interface support within the deep learning community. Instead, DANCE supports all types of data preprocessing and modeling across all modules including single modality, multimodality and spatial transcriptomics.

- **Deep Learning Infrastructure:** With the great increase in the number of single cells, classical methods [15, 68] cannot effectively enjoy the benefit

from big single-cell data, while deep learning has been proven to be effective. Furthermore, deep learning techniques are also good at handling high dimensional data, which is common for single-cell data. Unfortunately, the backend framework of the well-known Seurat is R, which limits its potential in the deep learning community due to restricted R interface support in the deep learning community. Scanpy only contains classical methodologies for downstream tasks. Recently scvi-tools [43] presents a Python library for deep probabilistic analysis of single-cell omics data. With 12 models, scvi-tools offers standardized access to 9 tasks. scvi-tools includes some deep learning methods but lacks the recent Graph Neural Networks (GNNs) based methods. In terms of models, scvi-tools selects baselines with a concentration on statistical models according to their supporting data protocol. As a comparison, DANCE is a comprehensive deep learning library of single-cell analysis. Popular deep learning infrastructures like Autoencoders [116] and GNNs are supported and applicable for all modules.

– **Standardized Benchmarks:** To the best of our knowledge, DANCE is the first comprehensive benchmark platform covering all modules in single-cell analysis. A few unique features have been developed to achieve this goal. We first collect task specific standard benchmark datasets and provide easy access to them by simply changing the parameter setting. Under each task, representative classical and deep learning algorithms are implemented as baselines. Those baselines are further fine-tuned on all collected benchmark datasets to reproduce similar or even better performance compared to original papers. To easily reproduce the results of our finetuned models, end users only need to run one command line where we wrap all super-parameters in advance to obtain reported performance.

		Scanpy	Seurat	scvi-tools	DeepCell	Squidpy	DANCE
Comprehensive Module Coverage	Single Modality	✓	✓	✓	✓	✗	✓
	Multimodality	✓	✓	✓	✗	✗	✓
	Spatial	✓	✓	✓	✗	✓	✓
Deep Learning Infrastructure	Classical Deep Learning	✗	✗	✓	✓	✓	✓
	GNNs	✗	✗	✗	✗	✓	✓
Standardized Benchmarks	Benchmark Datasets	✗	✗	✗	✓	✗	✓
	Task Specific Algorithms	✗	✗	✗	✓	✗	✓
	Reproducible Command Lines	✗	✗	✗	✗	✗	✓

Table 1: Comparison between DANCE and other popular single-cell libraries and toolkits.

One of the highlights of DANCE is the reproducibility of models. The diverse programming languages and backend frameworks of existing methods make systematic benchmark evaluation challenging for fair performance comparison. In such case, we implement all models in a unified development environment based on python language using Pytorch [105], Deep Graph Library (DGL) [141] and

PyTorch Geometric (PyG) [40] as backbone frameworks. In addition, we formulate all baselines into a generic fit-predict-score paradigm. From the reproducibility perspective, for each task, every implemented algorithm is fine-tuned on all collected standard benchmarks via grid search to get the best model, and the corresponding super-parameters are saved into only one command line for user's reproducibility. We also provide one example for each model as a reference.

2 Related Work

2.1 Single-Cell Analysis

Single Modality Next-generation sequencing allows for high-throughput transcriptome analysis. While bulk RNA sequencing data focuses on average gene expression profiles [76, 100, 110], single-cell RNA-sequencing (scRNA-seq) data quantifies gene expression at the cell level, offering unprecedented advantages to understanding biological mechanisms. In scRNA-seq experiments, single cells need to be isolated and captured; then, the captured cells go through lysis [66, 76, 100]. Reverse transcription is then applied to lysed cells for mRNA selection and cDNA synthesis, which is a key step that determines experiment sensitivity and the level of technical noise due to sampling noise, which is often assumed to have a Poisson distribution [58]. The rapid development of sequencing technologies has enabled highly scalable experiments where a massive number of cells could be processed simultaneously through microwell-based methods such as CytoSeq [58], microfluidics-based methods like Fluidigm C1 HT, or droplet-based methods like Drop-seq [86] and Chromium 10X [155]. In particular, the development of unique molecular identifiers (UMIs) improves the quality of scRNA-seq by barcoding each mRNA molecule within a cell individually during reverse transcription and thus mitigates PCR amplification bias. In addition to scRNA-sequencing, other types of single-cell sequencing assays also emerged recently. Two common branches are sequencing for surface protein such as CITE-seq [124] and sequencing for chromatin accessibility as represented by scATAC-seq [70].

Single-cell sequencing technology enables the simultaneous screening of thousands of genes in a massive number of cells and yields valuable insights in cell heterogeneity and functions. For instance, it has shed light on new understandings in oncological studies [72,88,145], cell characterizations [27,48,49,112,133,134,137], immune heterogeneity [44, 103, 154], and cell differentiation [12, 69, 97, 115]. In order to answer important biological questions, new algorithms are required that are specially devised for single-cell sequencing data. Unlike bulk sequencing, single-cell sequencing technologies profile thousands or even millions of cells, resulting in high-dimensional datasets. Furthermore, single-cell sequencing data suffers from high sparsity [71], i.e, a large proportion of genes with zero reads in the count matrix, a phenomenon known as dropout [8, 63, 66, 113]. Thus, new algorithms are needed to efficiently analyze single-cell sequencing data in order to tackle tasks such as data imputation, cell clustering, cell typing, and cell trajectory inferences.

Multimodality There have been exploding experiment technologies that obtain high-resolution features of single-cell, and many of them devoted to adapting assays of different omics to single-cell resolution (e.g., scDNA-seq [87], scATAC-seq [70], REAP-seq [108], etc)Recently, advances in single-cell technology have led to simultaneous assays of multi-omics in a single cell. For example, cellular indexing of transcriptome and epitopes by sequencing (CITE-seq) [124] simultaneously profiles mRNA gene expression and surface protein abundance; sci-CAR [16] and SNARE-seq [23] jointly measure mRNA gene expression and chromatin accessibility. Integrated analysis of multi-omics single-cell data has brought many important applications and achievements, such as revealing new cell populations [54] or regulatory networks [37, 59, 82, 153]. Current multi-omics analysis can be roughly categorized into two types, i.e., the joint analysis of data from different single-modal sources, and the analysis of data from multi-modal assays. Both of them can provide new insights into cell states. The most important problem in multimodal data analysis is how to integrate multimodal data to provide more accurate and in-depth cellular representation. To this end, integration algorithms can also be considered in two categories. The first category of methods [18, 37, 59, 61, 77, 122, 152, 153] focus on alignment between datasets, while the second category of methods [92, 143, 146, 149, 158] dedicate to capture cellular characterization from multimodality.

Although various successful downstream applications have demonstrated the effectiveness of multimodal data, it is still difficult to directly evaluate and compare the performance of different integration methods. One common way to evaluate those methods is to calculate Normalized Mutual Information (NMI) between the predicted cluster labels and predefined cell type labels. A more reasonable way to benchmark multimodal integration, as suggested in a recent work [80], is to leverage multi-omics aligned data to provide ground truth for multimodal integration, where two modalities are simultaneously measured in each cell (e.g. CITE-seq [124]). Three key tasks have been defined(i.e., modality prediction, modality matching, and joint embedding) to comprehensively evaluate the power of various integration methods. Modality prediction is to predict one modality from another. In modality matching, we aim to identify observations of the same cells among different modalities, while the ground truth correspondence is given by the dataset. Joint embedding is to directly evaluate the integrated embedding by comprehensive metrics based on biological states preservation and batch effects removal.

Spatial Transcriptomics While single-cell sequencing technologies are able to isolate cells from heterogeneous cell populations, they lose sight of the original microenvironments in the process. Spatial transcriptomics technologies are a more recent development in transcriptomics profiling. They are able to spatially resolve transcriptomics profiles. This allows researchers to further investigate and understand the spatial context of cells, and cell clusters [13]. However, the spatially resolved profiling regions for most technologies are not at single-cell

resolution, which motivates the problem of spatial (cell-type) deconvolution and segmentation.

Spatially resolved RNA profiling technologies can be roughly categorized as profiler-based and imager-based technologies [55]. Profiler-based technologies use Next-Generation Sequencing (NGS) readouts, and include Spatial Transcriptomics (10x Visium platform) [126], Slide-seqV2 [123], and Digital Spatial Profiling (Nanonstring GeoMx platform) [90]. The profiler-based technologies offer high-plex data, though not at single-cell resolution. Imager-based technologies apply a sequence of cyclic In situ hybridization (ISH) and imaging. Recent methods include MERFISH (Visgen MERSCOPE platform) [93], seqFISH+ [39], and the Spatial Molecular Imager (Nanostring CosMx platform) [55]. Imager-based technologies offer single-cell resolution (even sub-cellular), though usually not as high-plex as the profiler-based methods [55].

2.2 Graph Representation Learning

Graph representation learning has attracted increasing attention [84], since graph data are ubiquitous in the real world, e.g. social networks [98], knowledge graphs [60] and biological networks [106]. However, graph data has a more complex topology than text or image data with regular structures, causing it difficult to analyze. To facilitate downstream analysis, a natural idea is to denote each node with a vector while encoding intrinsic graph properties. This kind of methods is called graph embedding [84], including random-walk-based methods [47, 107], matrix-factorization-based methods [4, 10], and deep learning methods [147].

Random walk approaches (e.g., deepwalk [107], node2vec [47]) start with sampling the neighborhoods of nodes rather than directly using the global information of the entire graph. This allows the model to capture higher-order relationships between nodes, but they also lose some global structural information. Matrix factorization methods (e.g., graph laplacian eigenmaps [10], graph factorization [4]) have a solid mathematical foundation, but they are not scalable to large graphs, due to the high space complexity of proximity matrix construction and computational complexity of eigen decomposition. Moreover, most factorization methods only conserve the first-order proximity [20]. Deep learning models, also known as graph neural networks (e.g., GCN [64], GraphSAGE [50]), are generally achieving state-of-the-art performance in various applications. However, they are mathematically complex and thus lack interpretability. To be concrete, graph neural works (GNNs) iteratively propagate and transform node features to obtain node embeddings [46]. Therefore the node embeddings encode high-order topological structure information. In addition, GNNs provide denoising effects by smoothing graph signals through filtering eigenvalues of the graph Laplacian [83]. Because of these advantages, graph representation learning can be very effective for single-cell analysis. For instance, it can facilitate cell-cell interaction detection [73] and gene-network inference [59].

2.3 Deep Learning for Single Cell Analysis

In contrast to the bulk sampling techniques, single-cell sequencing techniques can produce millions of samples in a single experiment, which far exceeds the sample size of previous datasets. Along with this is the emergence of a large number of new research topics, mainly focusing on understanding the association between gene expressions and cellular behaviors. These changes have led researchers to focus on machine learning algorithms, which make good use of large training sets to optimize a given objective function. Particularly, deep learning methods consistently show outstanding performance in numerous machine learning applications [34, 111]. As a result, deep learning models are now widely used in single-cell analysis and have greatly aided the development of immunology, oncology, pharmacology, and many other disciplines [21, 45, 135].

The enormous potential of single-cell data comes with great complexity caused by various underlying biological and technical factors. To address this issue, several pre-processing steps are developed into the analysis pipeline. In the early stages, quality control and normalization are carefully designed. After that, complex machine learning tasks are introduced, e.g., batch effect correction, data imputation, and dimension reduction. Special types of single-cell data may require further processing, such as multimodal data integration for multi-omics and cell deconvolution for spatial transcriptomics. All these tasks can be facilitated by deep learning methods [6, 79, 114, 139]. Meanwhile, Deep learning methods [26,32,36,42,81,99,128,131,132,148,151] consistently outperform other classical machine learning techniques in downstream tasks, including clustering, cell type annotation, disease prediction, gene network inference, and trajectory analysis.

3 An Overview of DANCE

3.1 Environment Requirements and Setup

DANCE works on python ≥ 3.8 and Pytorch $\geq 1.11.0$. All dependencies are listed in Appendix A. After cloning this repository, run setup.py to install DANCE into the local python environment or install it directly from pip install as below:

```
pip install pydance
```

3.2 The Architecture Design

Figure 1 provides an overall design of the architecture of the DANCE package. The DANCE package consists of two key components: lower-level infrastructure and upper-level task development.

Lower-level Infrastructure From the hardware perspective, CPU running is supported for all methods developed in DANCE. In addition, for deep learning

based methods, we also support GPU running to accelerate the training process, especially for large-scale datasets. In the future, cluster running for deep learning methods would be also developed to support model training across multiple GPUs. The backbone framework in DANCE is Pytorch [105], which is used for high-performance deep learning model development. To support various methods for deep learning on graphs and other irregular structures, we take both Deep Graph Library (DGL) [141] and PyTorch Geometric (PyG) [40] as graph engines in DANCE. Various types of preprocessing functionalities are provided in the Transforms folder to process data before model training. For methods based on Graph Neural Networks (GNNs), we also support distinct ways of graph construction to convert cell-gene data like RNA sequencing (RNA-seq) to cell-cell, cell-gene and gene-gene graphs. What's more, spatial coordinates and image features of single cells can be also extracted to help construct graphs for spatial transcriptomics. Those lower-level interfaces are helpful for developers to build their models on downstream tasks without building "wheels" from scratch.

Upper-level Task Development Based on the infrastructure described above, individual modules and tasks can be further defined and developed. Currently, we support tasks under single modality profiling, multimodal profiling, and spatial transcriptomics modules, which correspond to three stages of single-cell technology development. Under each module, classic tasks are covered, and representative methods are implemented through the evaluation on several standard benchmarks. Note that upper-level task development is highly flexible and extensible. This indicates that users can readily extend their new modules, tasks, models, and datasets into the existing repository of DANCE.

3.3 A Pipeline of DANCE

Data Loading For each task, we have a generic interface to load datasets. All datasets supported by DANCE are cached on the cloud. Users don't have to download their interested datasets manually. They just need to specify an individual dataset when calling the data loader interface. For example, we can run graph-sc model on **10X PBMC** dataset for the clustering task using the following command line:

```
python graphsc.py --dataset='10X_PBMC'
```

Data Processing After data loading, a collection of data processing methods is provided before model training. They are divided into two parts: preprocessing and graph construction.

- **Preprocessing:** We provide rich preprocessing functions such as normalization, dimension reduction, gene filtering and so on. Take graph-sc model as an example, we filter out the rarely expressed genes and normalize the

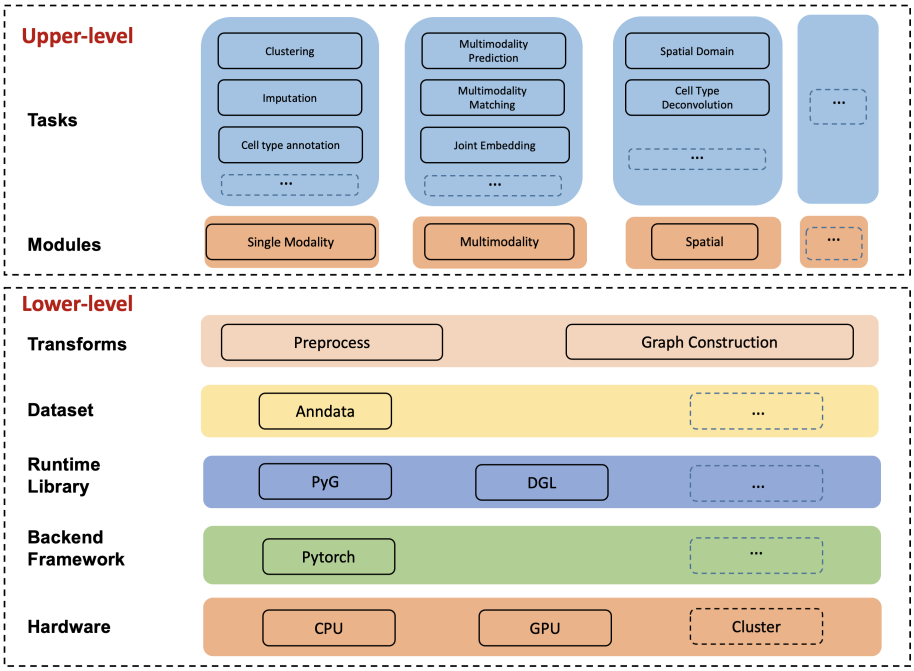

Fig. 1: The architecture of DANCE package.

remaining to obtain the same total count for each cell. Then only the highly expressed genes (top 3000 by default) are kept for clustering [26].

– **Graph Construction:** This is required for GNNs based method. Before model training, we have to convert data to graphs in preparation for graph operations. DANCE provides a variety of ways of graph construction. In graph-sc implementation, we construct a weighted heterogeneous cell-to-gene graph, where the types of nodes can be cell and gene nodes. The weight of each edge between each cell node and its corresponding gene node is determined by gene counts, and there is no edge linked between any pairs of cell or gene nodes.

Model Training and Evaluation All models in DANCE have generic interfaces for model training and evaluation. The unified interface for model training is **model.fit()** while that for model prediction is **model.predict()**, which returns the predictions of test data. Furthermore, **model.score()** acts as a generic interface to evaluate how well each model is. The metric of the score function depends on each task. Take graph-sc for an example, after fitting the model with chosen hyperparameters, we can access the performance of graph-sc by calling the score function, which will return ARI and NMI scores to indicate the quality of the clusters.

4 DANCE Benchmark: Modules, Tasks, Models, and Datasets

As shown in Figure 2, DANCE is capable of supporting modules of single modality, multimodality and spatial transcriptomics. Under each module, we benchmark several tasks with popular models across standard datasets. Here, we take the task of Clustering in the module of single modality as an example. Various types of methods are implemented including Graph Neural Networks based methods including graph-sc [26], scTAG [151] and scDSC [42] and AutoEncoders based methods including scDeepCluster [131] and scDCC [132]. To ensure a systematic evaluation and fair performance comparison of different models, several standard benchmark datasets such as 10X PBMC 4K [156], Mouse Bladder Cells [51], Worm Neuron Cells [17] and Mouse Embryonic Stem CellS [65] for the task are collected for evaluation. Currently, there are 3 modules, 8 tasks, 32 models and 21 datasets supported by DANCE. Please refer to Appendix D and Appendix C for more details about supported models and datasets, respectively.

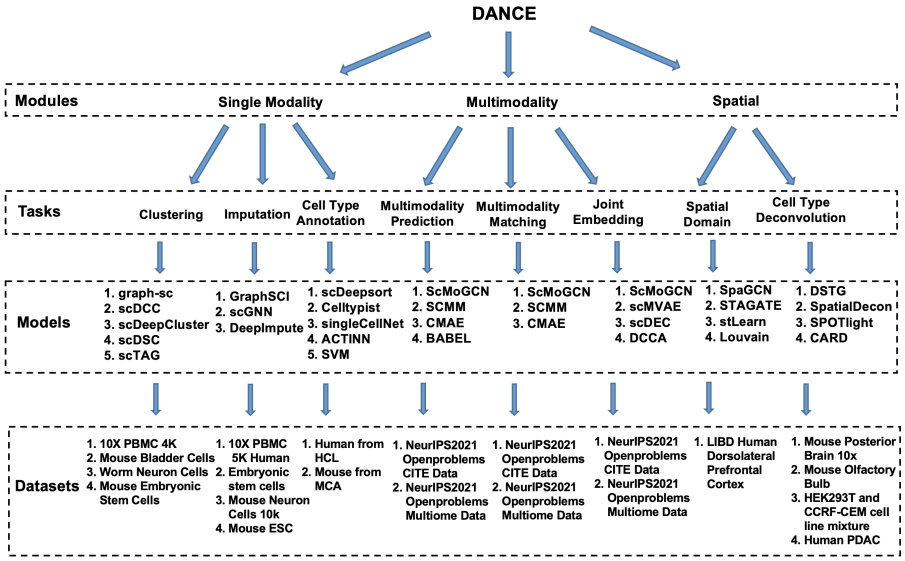

Fig. 2: A summary of modules, tasks, models and datasets supported by the DANCE package.

5 Benchmarking and Reproduction

One of the highlights of DANCE is the reproducibility and potential benchmarking. It is always challenging but helpful to reproduce the performance

of the published works in single-cell analysis since the implementations are based on different programming languages and different backend frameworks. To tackle this challenge, we implement all models based on Pytorch, PyG and DGL, and put all baselines into the fit-predict-score structure. For every task, we choose several benchmark datasets and tune all baselines on every dataset. When implementing baselines, we refer to the implementation details in the original GitHub repositories and the corresponding papers. For models without suggested parameters, we perform grid search and random search to obtain the best parameters. All the parameter settings can be found via `https://github.com/OmicsML/dance/tree/main/examples`, where we keep command line information for reproduction at the end of every example file.

To illustrate reproducibility and benchmarking in DANCE, we delve deeper into the cell type annotation task. Note that all reproductions are accessible on GitHub: `https://github.com/OmicsML/dance`. Currently, DANCE supports five models for this task. It includes scDeepsort [118] as a GNN-based method. ACTINN [81] and singleCellNet [128] are representative deep learning methods. We also cover support vector machine (SVM) and Celltypist [32] as traditional machine learning baselines. We select datasets on which all five models have reported results, i.e., MCA [52] for mouse and HCL [53] for human. As for the evaluation purpose, we pull three tissues from the MCA and pick one dataset from each tissue, i.e., Mouse Brain 2695, Mouse Spleen 1759 and Mouse Kidney 203.

Model	Mouse Brain 2695 (current / reported)	Mouse Spleen 1759 (current / reported)	Mouse Kidney 203 (current / reported)
scDeepsort	0.363 / 0.363	0.965 / 0.965	0.901 / 0.911
Celltypist*	0.680 / 0.666	0.966 / 0.848	0.879 / 0.832
singleCellNet	0.693 / 0.803	0.975 / 0.975	0.795 / 0.842
ACTINN	0.860 / 0.778	0.516 / 0.236	0.829 / 0.798
SVM	0.683 / 0.683	0.056 / 0.049	0.704 / 0.695

Table 2: Reproduced and reported ACC for the cell type annotation task.

*To meet the format requirements, we renormalize datasets before running the original implementation of Celltypist.

Table 2 demonstrates the reproduced and reported results of the cell-type annotation task in terms of accuracy (ACC). For every table cell, on the left, we have the reproduced annotation ACC based on DANCE, and on the right, we show the reported ACC in the original works. As shown in Table 2, SVM and scDeepsort [118] get similar results as reported. Impressively, Celltypist [32] and ACTINN [81] bring out advancement under the DANCE implementation with an average absolute increase of 7% and 44% over three datasets, respectively. Note that only singleCellNet [128] produces lower ACC compared to the reported results. Among all supported tasks, DANCE achieves uniform benchmarking grounded in Pytorch, which requires less configuration and allows users

to conduct analysis solely using Python. The structure of some models may be modified in the process of transformation from R or TensorFlow to Pytorch, which in consequence affects the reproduced results (e.g., singleCellNet).

6 Open-Source Contribution

DANCE is an open-source package, and everyone can contribute to this platform with extra tasks, models, and standard benchmark datasets by following the instructions below:

- **Codebase Structure Guidance:** Before contributing to DANCE, you need to understand the codebase structure in DANCE and make sure you are going to modify the correct files or put new files into the correct place. dance is the root of the package. datasets is the dataloader module, which contains task specific data loaders. modules is the model module, which covers all implemented algorithms. Each model is created as an individual file under the corresponding task folder. transforms is the data processing module, which deals with data preprocessing in preprocess and graph construction in graph_construct. You are flexible to pick up any existing functionalities in those two files for your model development and welcome to contribute new ones if your desired ones are not provided. examples is the example reference module, which presents one example for one model.
- **Code Style Guidance:** The contributed codes are required to follow the standard python coding style. For more details, please refer to Style Guide for Python Code.
- **Testing:** To ensure that your contributed code does not impact the normal operation of the existing codebase, you need to ensure that all tests pass before submitting. Please refer to the **Run Test** section in DANCE for how to run tests.
- **Reproducibility:** This is only required for contributed new models. An example file is necessary to present how to run your model on existing standard benchmarks. Furthermore, you need to place command lines at the end of the example file to obtain the best performance for reproducibility purposes. One command line corresponds to one standard benchmark dataset running of your contributed model.
- **Documentation:** Submitted code should be documented or commented on for easy readability purposes by users. Please refer to Numpy Style Docstring Guide for more details.

7 Impacts and Future Directions

In face of reproducibility issues of computational models in the field of single-cell analysis, we believe the DANCE platform will bring meaningful contributions to the whole single-cell community. To be specific, end users don't have to spend a lot of effort in implementing and tuning models. Instead, they only need to

run the command line we provide to easily reproduce results from the original paper. In addition, with our implementation, the performance of some models is even better than that reported in the original paper; we also provide GPUs support for the accelerated training purpose for deep learning based models in our implementation. It is worth noting that our DANCE package is an open-source package where every developer can contribute to advancing this field.

Since the functionalities of preprocessing and graph construction in current DANCE are not consolidated. It would be enhanced in the future. DANCE would be released as a SaaS service, which means that users are not limited to their local computation and storage resources. The interactive interface will be also provided for end users to use. In such case, after users upload their datasets to the platform, they only need to click on some buttons to select the preprocessing functions and the interested tasks and models for running without coding skills requirement. The results would be visualized on the website instantly. The leaderboard can be further developed to evaluate the efficacy and generalizability of state-of-the-art computational methods. From the data perspective, the collected datasets enable us to integrate them into atlas reference database via the exploration of integrating different modalities or even tissues. Later on, this atlas would be released for public access to further research in the whole single-cell community. Last but not least, AutoML and model explainability will be supported which can further minimize the efforts and ML background requirements for DANCE users.

References

1. Mouse olfactory bulb data. `https://www.10xgenomics.com/resources/datasets/adult-mouse-olfactory-bulb-1-standard-1`
2. Mouse posterior brain 10x visium data. `https://support.10xgenomics.com/spatial-gene-expression/datasets/1.0.0/V1_Mouse_Brain_Sagittal_Posterior`
3. Abdelaal, T., Michielsen, L., Cats, D., Hoogduin, D., Mei, H., Reinders, M.J., Mahfouz, A.: A comparison of automatic cell identification methods for single-cell rna sequencing data. Genome biology **20**(1), 1–19 (2019)
4. Ahmed, A., Shervashidze, N., Narayanamurthy, S., Josifovski, V., Smola, A.J.: Distributed large-scale natural graph factorization. In: Proceedings of the 22nd international conference on World Wide Web. pp. 37–48 (2013)
5. et al., L.: A sandbox for prediction and integration of DNA, RNA, and proteins in single cells. In: NeurIPS Datasets and Benchmarks Track (Round 2) (2021), Dataset Link: https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE194122
6. Arisdakessian, C., Poirion, O., Yunits, B., Zhu, X., Garmire, L.X.: DeepImpute: An accurate, fast, and scalable deep neural network method to impute single-cell RNA-seq data **20**(1), 211
7. Asp, M., Bergenstråhle, J., Lundeberg, J.: Spatially resolved transcriptomes—next generation tools for tissue exploration. BioEssays **42**(10), 1900221 (2020)

8. Bacher, R., Kendziorski, C.: Design and computational analysis of single-cell RNA-sequencing experiments **17**(1), 63. https://doi.org/10.1186/s13059-016-0927-y, https://genomebiology.biomedcentral.com/articles/10.1186/s13059-016-0927-y
9. Balakrishnan, V.: All about the dirac delta function (?). Resonance **8**(8), 48–58 (2003)
10. Belkin, M., Niyogi, P.: Laplacian eigenmaps for dimensionality reduction and data representation. Neural computation **15**(6), 1373–1396 (2003)
11. Blondel, V., Guillaume, J., Lambiotte, R., Lefebvre, E.: Fast unfolding of community hierarchies in large networks (2008)
12. Buenrostro, J.D., Corces, M.R., Lareau, C.A., Wu, B., Schep, A.N., Aryee, M.J., Majeti, R., Chang, H.Y., Greenleaf, W.J.: Integrated single-cell analysis maps the continuous regulatory landscape of human hematopoietic differentiation. Cell **173**(6), 1535–1548.e16 (2018). https://doi.org/https://doi.org/10.1016/j.cell.2018.03.074, https://www.sciencedirect.com/science/article/pii/S009286741830446X
13. Burgess, D.J.: Spatial transcriptomics coming of age. Nature Reviews Genetics **20**(6), 317–317 (2019)
14. Cadwell, C.R., Scala, F., Li, S., Livrizzi, G., Shen, S., Sandberg, R., Jiang, X., Tolias, A.S.: Multimodal profiling of single-cell morphology, electrophysiology, and gene expression using patch-seq. Nature protocols **12**(12), 2531–2553 (2017)
15. Calgaro, M., Romualdi, C., Waldron, L., Risso, D., Vitulo, N.: Assessment of statistical methods from single cell, bulk rna-seq, and metagenomics applied to microbiome data. Genome biology **21**(1), 1–31 (2020)
16. Cao, J., Cusanovich, D.A., Ramani, V., Aghamirzaie, D., Pliner, H.A., Hill, A.J., Daza, R.M., McFaline-Figueroa, J.L., Packer, J.S., Christiansen, L., et al.: Joint profiling of chromatin accessibility and gene expression in thousands of single cells. Science **361**(6409), 1380–1385 (2018)
17. Cao, J., Packer, J.S., Ramani, V., Cusanovich, D.A., Huynh, C., Daza, R., Qiu, X., Lee, C., Furlan, S.N., Steemers, F.J., et al.: Comprehensive single-cell transcriptional profiling of a multicellular organism. Science **357**(6352), 661–667 (2017), Dataset Link: http://atlas.gs.washington.edu/worm-rna/docs/
18. Cao, K., Bai, X., Hong, Y., Wan, L.: Unsupervised topological alignment for single-cell multi-omics integration. Bioinformatics **36**(Supplement_1), i48–i56 (2020)
19. Chen, E.I., Hewel, J., Felding-Habermann, B., Yates, J.R.: Large scale protein profiling by combination of protein fractionation and multidimensional protein identification technology (mudpit). Molecular & Cellular Proteomics **5**(1), 53–56 (2006)
20. Chen, F., Wang, Y.C., Wang, B., Kuo, C.C.J.: Graph representation learning: a survey. APSIPA Transactions on Signal and Information Processing **9** (2020)
21. Chen, H., Ye, F., Guo, G.: Revolutionizing immunology with single-cell rna sequencing. Cellular & molecular immunology **16**(3), 242–249 (2019)
22. Chen, K.H., Boettiger, A.N., Moffitt, J.R., Wang, S., Zhuang, X.: Spatially resolved, highly multiplexed rna profiling in single cells. Science **348**(6233), aaa6090 (2015)
23. Chen, S., Lake, B.B., Zhang, K.: High-throughput sequencing of the transcriptome and chromatin accessibility in the same cell. Nature biotechnology **37**(12), 1452–1457 (2019)

24. Cheow, L.F., Courtois, E.T., Tan, Y., Viswanathan, R., Xing, Q., Tan, R.Z., Tan, D.S., Robson, P., Loh, Y.H., Quake, S.R., et al.: Single-cell multimodal profiling reveals cellular epigenetic heterogeneity. Nature methods **13**(10), 833–836 (2016)
25. Chu, L.F., Leng, N., Zhang, J., Hou, Z., Mamott, D., Vereide, D.T., Choi, J., Kendziorski, C., Stewart, R., Thomson, J.A.: Single-cell RNA-seq reveals novel regulators of human embryonic stem cell differentiation to definitive endoderm. Genome Biology **17**(1), 173 (Aug 2016). https://doi.org/10.1186/s13059-016-1033-x, Dataset Link: https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE75748
26. Ciortan, M., Defrance, M.: Gnn-based embedding for clustering scrna-seq data. Bioinformatics **38**(4), 1037–1044 (2022)
27. Crinier, A., Milpied, P., Escalière, B., Piperoglou, C., Galluso, J., Balsamo, A., Spinelli, L., Cervera-Marzal, I., Ebbo, M., Girard-Madoux, M., Jaeger, S., Bollon, E., Hamed, S., Hardwigsen, J., Ugolini, S., Vély, F., Narni-Mancinelli, E., Vivier, E.: High-dimensional single-cell analysis identifies organ-specific signatures and conserved NK cell subsets in humans and mice. Immunity **49**(5), 971–986.e5 (Nov 2018). https://doi.org/10.1016/j.immuni.2018.09.009, `https://doi.org/10.1016/j.immuni.2018.09.009`
28. Crosetto, N., Bienko, M., Van Oudenaarden, A.: Spatially resolved transcriptomics and beyond. Nature Reviews Genetics **16**(1), 57–66 (2015)
29. Danaher, P., Kim, Y., Nelson, B., Griswold, M., Yang, Z., Piazza, E., Beechem, J.M.: Advances in mixed cell deconvolution enable quantification of cell types in spatial transcriptomic data. Nature communications **13**(1), 1–13 (2022), Dataset Link: https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE174746
30. Danaher, P., Kim, Y., Nelson, B., Griswold, M., Yang, Z., Piazza, E., Beechem, J.M.: Advances in mixed cell deconvolution enable quantification of cell types in spatial transcriptomic data. Nature communications **13**(1), 1–13 (2022)
31. van Dijk, D., Sharma, R., Nainys, J., Yim, K., Kathail, P., Carr, A.J., Burdziak, C., Moon, K.R., Chaffer, C.L., Pattabiraman, D., Bierie, B., Mazutis, L., Wolf, G., Krishnaswamy, S., Pe'er, D.: Recovering Gene Interactions from Single-Cell Data Using Data Diffusion **174**(3), 716–729.e27. https://doi.org/10.1016/j.cell.2018.05.061, `https://linkinghub.elsevier.com/retrieve/pii/S0092867418307244`
32. Domínguez Conde, C., Xu, C., Jarvis, L., Rainbow, D., Wells, S., Gomes, T., Howlett, S., Suchanek, O., Polanski, K., King, H., et al.: Cross-tissue immune cell analysis reveals tissue-specific features in humans. Science **376**(6594), eabl5197 (2022)
33. Dong, K., Zhang, S.: Deciphering spatial domains from spatially resolved transcriptomics with an adaptive graph attention auto-encoder. Nature communications **13**(1), 1–12 (2022)
34. Dong, S., Wang, P., Abbas, K.: A survey on deep learning and its applications. Computer Science Review **40**, 100379 (2021)
35. Du, J., Zhang, S., Wu, G., Moura, J.M., Kar, S.: Topology adaptive graph convolutional networks. arXiv preprint arXiv:1710.10370 (2017)
36. Du, J.H., Gao, M., Wang, J.: Model-based trajectory inference for single-cell rna sequencing using deep learning with a mixture prior. bioRxiv (2020)
37. Duren, Z., Chen, X., Zamanighomi, M., Zeng, W., Satpathy, A.T., Chang, H.Y., Wang, Y., Wong, W.H.: Integrative analysis of single-cell genomics data by coupled nonnegative matrix factorizations. Proceedings of the National Academy of Sciences **115**(30), 7723–7728 (2018)

38. Elosua-Bayes, M., Nieto, P., Mereu, E., Gut, I., Heyn, H.: Spotlight: seeded nmf regression to deconvolute spatial transcriptomics spots with single-cell transcriptomes. Nucleic Acids Research **49**(9), e50–e50 (02 2021)
39. Eng, C.H.L., Lawson, M., Zhu, Q., Dries, R., Koulena, N., Takei, Y., Yun, J., Cronin, C., Karp, C., Yuan, G.C., Cai, L.: Transcriptome-scale super-resolved imaging in tissues by rna seqfish+. Nature (2019). https://doi.org/https://doi.org/10.1038/s41586-019-1049-y
40. Fey, M., Lenssen, J.E.: Fast graph representation learning with pytorch geometric. arXiv preprint arXiv:1903.02428 (2019)
41. Fujii, K., Nakano, T., Kawamura, T., Usui, F., Bando, Y., Wang, R., Nishimura, T.: Multidimensional protein profiling technology and its application to human plasma proteome. Journal of Proteome Research **3**(4), 712–718 (2004)
42. Gan, Y., Huang, X., Zou, G., Zhou, S., Guan, J.: Deep structural clustering for single-cell rna-seq data jointly through autoencoder and graph neural network. Briefings in Bioinformatics **23**(2), bbac018 (2022)
43. Gayoso, A., Lopez, R., Xing, G., Boyeau, P., Valiollah Pour Amiri, V., Hong, J., Wu, K., Jayasuriya, M., Mehlman, E., Langevin, M., et al.: A python library for probabilistic analysis of single-cell omics data. Nature Biotechnology **40**(2), 163–166 (2022)
44. Giladi, A., Amit, I.: Single-cell genomics: A stepping stone for future immunology discoveries. Cell **172**(1-2), 14–21 (Jan 2018). https://doi.org/10.1016/j.cell.2017.11.011, `https://doi.org/10.1016/j.cell.2017.11.011`
45. Giladi, A., Amit, I.: Single-cell genomics: a stepping stone for future immunology discoveries. Cell **172**(1-2), 14–21 (2018)
46. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017. Proceedings of Machine Learning Research (2017)
47. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 855–864 (2016)
48. Grün, D., Lyubimova, A., Kester, L., Wiebrands, K., Basak, O., Sasaki, N., Clevers, H., van Oudenaarden, A.: Single-cell messenger RNA sequencing reveals rare intestinal cell types. Nature **525**(7568), 251–255 (Aug 2015). https://doi.org/10.1038/nature14966, `https://doi.org/10.1038/nature14966`
49. Guo, X., Zhang, Y., Zheng, L., Zheng, C., Song, J., Zhang, Q., Kang, B., Liu, Z., Jin, L., Xing, R., Gao, R., Zhang, L., Dong, M., Hu, X., Ren, X., Kirchhoff, D., Roider, H.G., Yan, T., Zhang, Z.: Global characterization of t cells in non-small-cell lung cancer by single-cell sequencing. Nature Medicine **24**(7), 978–985 (Jun 2018). https://doi.org/10.1038/s41591-018-0045-3, `https://doi.org/10.1038/s41591-018-0045-3`
50. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. Advances in neural information processing systems **30** (2017)
51. Han, X., Wang, R., Zhou, Y., Fei, L., Sun, H., Lai, S., Saadatpour, A., Zhou, Z., Chen, H., Ye, F., et al.: Mapping the mouse cell atlas by microwell-seq. Cell **172**(5), 1091–1107 (2018), Dataset Link: https://figshare.com/s/865e694ad06d5857db4b
52. Han, X., Wang, R., Zhou, Y., Fei, L., Sun, H., Lai, S., Saadatpour, A., Zhou, Z., Chen, H., Ye, F., et al.: Mapping the mouse cell atlas by microwell-seq.

Cell **172**(5), 1091–1107 (2018), https://github.com/ZJUFanLab/scDeepSort/releases/download/Pre_processed_data/mouse_cell_atlas.7z
53. Han, X., Zhou, Z., Fei, L., Sun, H., Wang, R., Chen, Y., Chen, H., Wang, J., Tang, H., Ge, W., et al.: Construction of a human cell landscape at single-cell level. Nature **581**(7808), 303–309 (2020), https://github.com/ZJUFanLab/scDeepSort/releases/download/Pre/_processed/_data/human/_cell/_atlas.7z
54. Hao, Y., Hao, S., Andersen-Nissen, E., Mauck III, W.M., Zheng, S., Butler, A., Lee, M.J., Wilk, A.J., Darby, C., Zager, M., et al.: Integrated analysis of multimodal single-cell data. Cell **184**(13), 3573–3587 (2021)
55. He, S., Bhatt, R., Brown, C., Brown, E.A., Buhr, D.L., Chantranuvatana, K., Danaher, P., Dunaway, D., Garrison, R.G., Geiss, G., Gregory, M.T., Hoang, M.L., Khafizov, R., Killingbeck, E.E., Kim, D., Kim, T.K., Kim, Y., Klock, A., Korukonda, M., Kutchma, A., Lewis, Z.R., Liang, Y., Nelson, J.S., Ong, G.T., Perillo, E.P., Phan, J.C., Phan-Everson, T., Piazza, E., Rane, T., Reitz, Z., Rhodes, M., Rosenbloom, A., Ross, D., Sato, H., Wardhani, A.W., Williams-Wietzikoski, C.A., Wu, L., Beechem, J.M.: High-plex multiomic analysis in ffpe at subcellular level by spatial molecular imaging. bioRxiv (2022). https://doi.org/10.1101/2021.11.03.467020
56. Hou, W., Ji, Z., Ji, H., Hicks, S.C.: A systematic evaluation of single-cell RNA-sequencing imputation methods **21**(1), 218
57. Hu, J., Li, X., Coleman, K., Schroeder, A., Ma, N., Irwin, D.J., Lee, E.B., Shinohara, R.T., Li, M.: Spagcn: Integrating gene expression, spatial location and histology to identify spatial domains and spatially variable genes by graph convolutional network. Nature methods **18**(11), 1342–1351 (2021)
58. Islam, S., Zeisel, A., Joost, S., La Manno, G., Zajac, P., Kasper, M., Lönnerberg, P., Linnarsson, S.: Quantitative single-cell RNA-seq with unique molecular identifiers **11**(2), 163–166. https://doi.org/10.1038/nmeth.2772, http://www.nature.com/articles/nmeth.2772
59. Jansen, C., Ramirez, R.N., El-Ali, N.C., Gomez-Cabrero, D., Tegner, J., Merkenschlager, M., Conesa, A., Mortazavi, A.: Building gene regulatory networks from scatac-seq and scrna-seq using linked self organizing maps. PLoS computational biology **15**(11), e1006555 (2019)
60. Ji, S., Pan, S., Cambria, E., Marttinen, P., Philip, S.Y.: A survey on knowledge graphs: Representation, acquisition, and applications. IEEE Transactions on Neural Networks and Learning Systems **33**(2), 494–514 (2021)
61. Jin, S., Zhang, L., Nie, Q.: scai: an unsupervised approach for the integrative analysis of parallel single-cell transcriptomic and epigenomic profiles. Genome biology **21**(1), 1–19 (2020)
62. Kashyap, V., Sitalaximi, T., Chattopadhyay, P., Trivedi, R.: Dna profiling technologies in forensic analysis. International Journal of Human Genetics **4**(1), 11–30 (2004)
63. Kharchenko, P.V., Silberstein, L., Scadden, D.T.: Bayesian approach to single-cell differential expression analysis **11**(7), 740–742. https://doi.org/10.1038/nmeth.2967, http://www.nature.com/articles/nmeth.2967
64. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
65. Klein, A.M., Mazutis, L., Akartuna, I., Tallapragada, N., Veres, A., Li, V., Peshkin, L., Weitz, D.A., Kirschner, M.W.: Droplet barcoding for single-cell transcriptomics applied to embryonic stem cells. Cell **161**(5), 1187–1201 (2015), Dataset Link: https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE65525

66. Kolodziejczyk, A.A., Kim, J.K., Svensson, V., Marioni, J.C., Teichmann, S.A.: The Technology and Biology of Single-Cell RNA Sequencing **58**(4), 610–620. https://doi.org/10.1016/j.molcel.2015.04.005, https://linkinghub.elsevier.com/retrieve/pii/S1097276515002610
67. Kolodziejczyk, A.A., Kim, J.K., Svensson, V., Marioni, J.C., Teichmann, S.A.: The technology and biology of single-cell rna sequencing. Molecular cell **58**(4), 610–620 (2015)
68. Korthauer, K.D., Chu, L.F., Newton, M.A., Li, Y., Thomson, J., Stewart, R., Kendziorski, C.: A statistical approach for identifying differential distributions in single-cell rna-seq experiments. Genome biology **17**(1), 1–15 (2016)
69. Kowalczyk, M.S., Tirosh, I., Heckl, D., Rao, T.N., Dixit, A., Haas, B.J., Schneider, R.K., Wagers, A.J., Ebert, B.L., Regev, A.: Single-cell RNA-seq reveals changes in cell cycle and differentiation programs upon aging of hematopoietic stem cells. Genome Research **25**(12), 1860–1872 (Oct 2015). https://doi.org/10.1101/gr.192237.115, https://doi.org/10.1101/gr.192237.115
70. Labib, M., Kelley, S.O.: Single-cell analysis targeting the proteome. Nature Reviews Chemistry **4**(3), 143–158 (2020)
71. Lähnemann, D., Köster, J., Szczurek, E., McCarthy, D.J., Hicks, S.C., Robinson, M.D., Vallejos, C.A., Campbell, K.R., Beerenwinkel, N., Mahfouz, A., Pinello, L., Skums, P., Stamatakis, A., Attolini, C.S.O., Aparicio, S., Baaijens, J., Balvert, M., de Barbanson, B., Cappuccio, A., Corleone, G., Dutilh, B.E., Florescu, M., Guryev, V., Holmer, R., Jahn, K., Lobo, T.J., Keizer, E.M., Khatri, I., Kielbasa, S.M., Korbel, J.O., Kozlov, A.M., Kuo, T.H., Lelieveldt, B.P., Mandoiu, I.I., Marioni, J.C., Marschall, T., Mölder, F., Niknejad, A., Raczkowski, L., Reinders, M., de Ridder, J., Saliba, A.E., Somarakis, A., Stegle, O., Theis, F.J., Yang, H., Zelikovsky, A., McHardy, A.C., Raphael, B.J., Shah, S.P., Schönhuth, A.: Eleven grand challenges in single-cell data science **21**(1), 31. https://doi.org/10.1186/s13059-020-1926-6, https://genomebiology.biomedcentral.com/articles/10.1186/s13059-020-1926-6
72. Lei, Y., Tang, R., Xu, J., Wang, W., Zhang, B., Liu, J., Yu, X., Shi, S.: Applications of single-cell sequencing in cancer research: progress and perspectives. Journal of Hematology &amp Oncology **14**(1) (Jun 2021). https://doi.org/10.1186/s13045-021-01105-2, https://doi.org/10.1186/s13045-021-01105-2
73. Li, H., Ma, T., Hao, M., Wei, L., Zhang, X.: Decoding functional cell-cell communication events by multi-view graph learning on spatial transcriptomics. bioRxiv (2022)
74. Li, N., Overkleeft, H.S., Florea, B.I.: Activity-based protein profiling: an enabling technology in chemical biology research. Current opinion in chemical biology **16**(1-2), 227–233 (2012)
75. Li, W.V., Li, J.J.: An accurate and robust imputation method scImpute for single-cell RNA-seq data **9**(1), 997
76. Li, X., Wang, C.Y.: From bulk, single-cell to spatial RNA sequencing **13**(1), 36. https://doi.org/10.1038/s41368-021-00146-0, https://www.nature.com/articles/s41368-021-00146-0
77. Liu, J., Gao, C., Sodicoff, J., Kozareva, V., Macosko, E.Z., Welch, J.D.: Jointly defining cell types from multiple single-cell datasets using liger. Nature protocols **15**(11), 3632–3662 (2020)

78. Liu, Q., Chen, S., Jiang, R., Wong, W.H.: Simultaneous deep generative modelling and clustering of single-cell genomic data. Nature machine intelligence **3**(6), 536–544 (2021)
79. Lopez, R., Regier, J., Cole, M.B., Jordan, M.I., Yosef, N.: Deep generative modeling for single-cell transcriptomics. Nature methods **15**(12), 1053–1058 (2018)
80. Luecken, M.D., Burkhardt, D.B., Cannoodt, R., Lance, C., Agrawal, A., Aliee, H., Chen, A.T., Deconinck, L., Detweiler, A.M., Granados, A.A., et al.: A sandbox for prediction and integration of dna, rna, and proteins in single cells. In: Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2) (2021)
81. Ma, F., Pellegrini, M.: Actinn: automated identification of cell types in single cell rna sequencing. Bioinformatics **36**(2), 533–538 (2020)
82. Ma, S., Zhang, B., LaFave, L.M., Earl, A.S., Chiang, Z., Hu, Y., Ding, J., Brack, A., Kartha, V.K., Tay, T., et al.: Chromatin potential identified by shared single-cell profiling of rna and chromatin. Cell **183**(4), 1103–1116 (2020)
83. Ma, Y., Liu, X., Zhao, T., Liu, Y., Tang, J., Shah, N.: A unified view on graph neural networks as graph signal denoising. In: Proceedings of the 30th ACM International Conference on Information & Knowledge Management. pp. 1202–1211 (2021)
84. Ma, Y., Tang, J.: Deep learning on graphs. Cambridge University Press (2021)
85. Ma, Y., Zhou, X.: Spatially informed cell-type deconvolution for spatial transcriptomics. Nature Biotechnology pp. 1–11 (2022)
86. Macosko, E.Z., Basu, A., Satija, R., Nemesh, J., Shekhar, K., Goldman, M., Tirosh, I., Bialas, A.R., Kamitaki, N., Martersteck, E.M., Trombetta, J.J., Weitz, D.A., Sanes, J.R., Shalek, A.K., Regev, A., McCarroll, S.A.: Highly Parallel Genome-wide Expression Profiling of Individual Cells Using Nanoliter Droplets **161**(5), 1202–1214. https://doi.org/10.1016/j.cell.2015.05.002
87. Mallory, X.F., Edrisi, M., Navin, N., Nakhleh, L.: Methods for copy number aberration detection from single-cell dna-sequencing data. Genome biology **21**(1), 1–22 (2020)
88. Malta, T.M., Sokolov, A., Gentles, A.J., Burzykowski, T., Poisson, L., Weinstein, J.N., Kamińska, B., Huelsken, J., Omberg, L., Gevaert, O., Colaprico, A., Czerwińska, P., Mazurek, S., Mishra, L., Heyn, H., Krasnitz, A., Godwin, A.K., Lazar, A.J., Stuart, J.M., Hoadley, K.A., Laird, P.W., Noushmehr, H., Wiznerowicz, M., Caesar-Johnson, S.J., Demchok, J.A., Felau, I., Kasapi, M., Ferguson, M.L., Hutter, C.M., Sofia, H.J., Tarnuzzer, R., Wang, Z., Yang, L., Zenklusen, J.C., Zhang, J.J., Chudamani, S., Liu, J., Lolla, L., Naresh, R., Pihl, T., Sun, Q., Wan, Y., Wu, Y., Cho, J., DeFreitas, T., Frazer, S., Gehlenborg, N., Getz, G., Heiman, D.I., Kim, J., Lawrence, M.S., Lin, P., Meier, S., Noble, M.S., Saksena, G., Voet, D., Zhang, H., Bernard, B., Chambwe, N., Dhankani, V., Knijnenburg, T., Kramer, R., Leinonen, K., Liu, Y., Miller, M., Reynolds, S., Shmulevich, I., Thorsson, V., Zhang, W., Akbani, R., Broom, B.M., Hegde, A.M., Ju, Z., Kanchi, R.S., Korkut, A., Li, J., Liang, H., Ling, S., Liu, W., Lu, Y., Mills, G.B., Ng, K.S., Rao, A., Ryan, M., Wang, J., Weinstein, J.N., Zhang, J., Abeshouse, A., Armenia, J., Chakravarty, D., Chatila, W.K., de Bruijn, I., Gao, J., Gross, B.E., Heins, Z.J., Kundra, R., La, K., Ladanyi, M., Luna, A., Nissan, M.G., Ochoa, A., Phillips, S.M., Reznik, E., Sanchez-Vega, F., Sander, C., Schultz, N., Sheridan, R., Sumer, S.O., Sun, Y., Taylor, B.S., Wang, J., Zhang, H., Anur, P., Peto, M., Spellman, P., Benz, C., Stuart, J.M., Wong, C.K., Yau, C., Hayes, D.N., Parker, J.S., Wilkerson, M.D., Ally, A., Balasundaram, M., Bowlby, R., Brooks, D., Carlsen, R.,

Chuah, E., Dhalla, N., Holt, R., Jones, S.J., Kasaian, K., Lee, D., Ma, Y., Marra, M.A., Mayo, M., Moore, R.A., Mungall, A.J., Mungall, K., Robertson, A.G., Sadeghi, S., Schein, J.E., Sipahimalani, P., Tam, A., Thiessen, N., Tse, K., Wong, T., Berger, A.C., Beroukhim, R., Cherniack, A.D., Cibulskis, C., Gabriel, S.B., Gao, G.F., Ha, G., Meyerson, M., Schumacher, S.E., Shih, J., Kucherlapati, M.H., Kucherlapati, R.S., Baylin, S., Cope, L., Danilova, L., Bootwalla, M.S., Lai, P.H., Maglinte, D.T., Berg, D.J.V.D., Weisenberger, D.J., Auman, J.T., Balu, S., Bodenheimer, T., Fan, C., Hoadley, K.A., Hoyle, A.P., Jefferys, S.R., Jones, C.D., Meng, S., Mieczkowski, P.A., Mose, L.E., Perou, A.H., Perou, C.M., Roach, J., Shi, Y., Simons, J.V., Skelly, T., Soloway, M.G., Tan, D., Veluvolu, U., Fan, H., Hinoue, T., Laird, P.W., Shen, H., Zhou, W., Bellair, M., Chang, K., Covington, K., Creighton, C.J., Dinh, H., Doddapaneni, H., Donehower, L.A., Drummond, J., Gibbs, R.A., Glenn, R., Hale, W., Han, Y., Hu, J., Korchina, V., Lee, S., Lewis, L., Li, W., Liu, X., Morgan, M., Morton, D., Muzny, D., Santibanez, J., Sheth, M., Shinbrot, E., Wang, L., Wang, M., Wheeler, D.A., Xi, L., Zhao, F., Hess, J., Appelbaum, E.L., Bailey, M., Cordes, M.G., Ding, L., Fronick, C.C., Fulton, L.A., Fulton, R.S., Kandoth, C., Mardis, E.R., McLellan, M.D., Miller, C.A., Schmidt, H.K., Wilson, R.K., Crain, D., Curley, E., Gardner, J., Lau, K., Mallery, D., Morris, S., Paulauskis, J., Penny, R., Shelton, C., Shelton, T., Sherman, M., Thompson, E., Yena, P., Bowen, J., Gastier-Foster, J.M., Gerken, M., Leraas, K.M., Lichtenberg, T.M., Ramirez, N.C., Wise, L., Zmuda, E., Corcoran, N., Costello, T., Hovens, C., Carvalho, A.L., de Carvalho, A.C., Fregnani, J.H., Longatto-Filho, A., Reis, R.M., Scapulatempo-Neto, C., Silveira, H.C., Vidal, D.O., Burnette, A., Eschbacher, J., Hermes, B., Noss, A., Singh, R., Anderson, M.L., Castro, P.D., Ittmann, M., Huntsman, D., Kohl, B., Le, X., Thorp, R., Andry, C., Duffy, E.R., Lyadov, V., Paklina, O., Setdikova, G., Shabunin, A., Tavobilov, M., McPherson, C., Warnick, R., Berkowitz, R., Cramer, D., Feltmate, C., Horowitz, N., Kibel, A., Muto, M., Raut, C.P., Malykh, A., Barnholtz-Sloan, J.S., Barrett, W., Devine, K., Fulop, J., Ostrom, Q.T., Shimmel, K., Wolinsky, Y., Sloan, A.E., Rose, A.D., Giuliante, F., Goodman, M., Karlan, B.Y., Hagedorn, C.H., Eckman, J., Harr, J., Myers, J., Tucker, K., Zach, L.A., Deyarmin, B., Hu, H., Kvecher, L., Larson, C., Mural, R.J., Somiari, S., Vicha, A., Zelinka, T., Bennett, J., Iacocca, M., Rabeno, B., Swanson, P., Latour, M., Lacombe, L., Têtu, B., Bergeron, A., McGraw, M., Staugaitis, S.M., Chabot, J., Hibshoosh, H., Sepulveda, A., Su, T., Wang, T., Potapova, O., Voronina, O., Desjardins, L., Mariani, O., Roman-Roman, S., Sastre, X., Stern, M.H., Cheng, F., Signoretti, S., Berchuck, A., Bigner, D., Lipp, E., Marks, J., McCall, S., McLendon, R., Secord, A., Sharp, A., Behera, M., Brat, D.J., Chen, A., Delman, K., Force, S., Khuri, F., Magliocca, K., Maithel, S., Olson, J.J., Owonikoko, T., Pickens, A., Ramalingam, S., Shin, D.M., Sica, G., Meir, E.G.V., Zhang, H., Eijckenboom, W., Gillis, A., Korpershoek, E., Looijenga, L., Oosterhuis, W., Stoop, H., van Kessel, K.E., Zwarthoff, E.C., Calatozzolo, C., Cuppini, L., Cuzzubbo, S., DiMeco, F., Finocchiaro, G., Mattei, L., Perin, A., Pollo, B., Chen, C., Houck, J., Lohavanichbutr, P., Hartmann, A., Stoehr, C., Stoehr, R., Taubert, H., Wach, S., Wullich, B., Kycler, W., Murawa, D., Wiznerowicz, M., Chung, K., Edenfield, W.J., Martin, J., Baudin, E., Bubley, G., Bueno, R., Rienzo, A.D., Richards, W.G., Kalkanis, S., Mikkelsen, T., Noushmehr, H., Scarpace, L., Girard, N., Aymerich, M., Campo, E., Giné, E., Guillermo, A.L., Bang, N.V., Hanh, P.T., Phu, B.D., Tang, Y., Colman, H., Evason, K., Dottino, P.R., Martignetti, J.A., Gabra, H., Juhl, H., Akeredolu, T., Stepa, S., Hoon, D., Ahn, K., Kang, K.J., Beuschlein, F., Breggia, A.,

Birrer, M., Bell, D., Borad, M., Bryce, A.H., Castle, E., Chandan, V., Cheville, J., Copland, J.A., Farnell, M., Flotte, T., Giama, N., Ho, T., Kendrick, M., Kocher, J.P., Kopp, K., Moser, C., Nagorney, D., O'Brien, D., O'Neill, B.P., Patel, T., Petersen, G., Que, F., Rivera, M., Roberts, L., Smallridge, R., Smyrk, T., Stanton, M., Thompson, R.H., Torbenson, M., Yang, J.D., Zhang, L., Brimo, F., Ajani, J.A., Gonzalez, A.M.A., Behrens, C., Bondaruk, J., Broaddus, R., Czerniak, B., Esmaeli, B., Fujimoto, J., Gershenwald, J., Guo, C., Lazar, A.J., Logothetis, C., Meric-Bernstam, F., Moran, C., Ramondetta, L., Rice, D., Sood, A., Tamboli, P., Thompson, T., Troncoso, P., Tsao, A., Wistuba, I., Carter, C., Haydu, L., Hersey, P., Jakrot, V., Kakavand, H., Kefford, R., Lee, K., Long, G., Mann, G., Quinn, M., Saw, R., Scolyer, R., Shannon, K., Spillane, A., Stretch, J., Synott, M., Thompson, J., Wilmott, J., Al-Ahmadie, H., Chan, T.A., Ghossein, R., Gopalan, A., Levine, D.A., Reuter, V., Singer, S., Singh, B., Tien, N.V., Broudy, T., Mirsaidi, C., Nair, P., Drwiega, P., Miller, J., Smith, J., Zaren, H., Park, J.W., Hung, N.P., Kebebew, E., Linehan, W.M., Metwalli, A.R., Pacak, K., Pinto, P.A., Schiffman, M., Schmidt, L.S., Vocke, C.D., Wentzensen, N., Worrell, R., Yang, H., Moncrieff, M., Goparaju, C., Melamed, J., Pass, H., Botnariuc, N., Caraman, I., Cernat, M., Chemencedji, I., Clipca, A., Doruc, S., Gorincioi, G., Mura, S., Pirtac, M., Stancul, I., Tcaciuc, D., Albert, M., Alexopoulou, I., Arnaout, A., Bartlett, J., Engel, J., Gilbert, S., Parfitt, J., Sekhon, H., Thomas, G., Rassl, D.M., Rintoul, R.C., Bifulco, C., Tamakawa, R., Urba, W., Hayward, N., Timmers, H., Antenucci, A., Facciolo, F., Grazi, G., Marino, M., Merola, R., de Krijger, R., Gimenez-Roqueplo, A.P., Piché, A., Chevalier, S., McKercher, G., Birsoy, K., Barnett, G., Brewer, C., Farver, C., Naska, T., Pennell, N.A., Raymond, D., Schilero, C., Smolenski, K., Williams, F., Morrison, C., Borgia, J.A., Liptay, M.J., Pool, M., Seder, C.W., Junker, K., Omberg, L., Dinkin, M., Manikhas, G., Alvaro, D., Bragazzi, M.C., Cardinale, V., Carpino, G., Gaudio, E., Chesla, D., Cottingham, S., Dubina, M., Moiseenko, F., Dhanasekaran, R., Becker, K.F., Janssen, K.P., Slotta-Huspenina, J., Abdel-Rahman, M.H., Aziz, D., Bell, S., Cebulla, C.M., Davis, A., Duell, R., Elder, J.B., Hilty, J., Kumar, B., Lang, J., Lehman, N.L., Mandt, R., Nguyen, P., Pilarski, R., Rai, K., Schoenfield, L., Senecal, K., Wakely, P., Hansen, P., Lechan, R., Powers, J., Tischler, A., Grizzle, W.E., Sexton, K.C., Kastl, A., Henderson, J., Porten, S., Waldmann, J., Fassnacht, M., Asa, S.L., Schadendorf, D., Couce, M., Graefen, M., Huland, H., Sauter, G., Schlomm, T., Simon, R., Tennstedt, P., Olabode, O., Nelson, M., Bathe, O., Carroll, P.R., Chan, J.M., Disaia, P., Glenn, P., Kelley, R.K., Landen, C.N., Phillips, J., Prados, M., Simko, J., Smith-McCune, K., VandenBerg, S., Roggin, K., Fehrenbach, A., Kendler, A., Sifri, S., Steele, R., Jimeno, A., Carey, F., Forgie, I., Mannelli, M., Carney, M., Hernandez, B., Campos, B., Herold-Mende, C., Jungk, C., Unterberg, A., von Deimling, A., Bossler, A., Galbraith, J., Jacobus, L., Knudson, M., Knutson, T., Ma, D., Milhem, M., Sigmund, R., Godwin, A.K., Madan, R., Rosenthal, H.G., Adebamowo, C., Adebamowo, S.N., Boussioutas, A., Beer, D., Giordano, T., Mes-Masson, A.M., Saad, F., Bocklage, T., Landrum, L., Mannel, R., Moore, K., Moxley, K., Postier, R., Walker, J., Zuna, R., Feldman, M., Valdivieso, F., Dhir, R., Luketich, J., Pinero, E.M.M., Quintero-Aguilo, M., Carlotti, C.G., Santos, J.S.D., Kemp, R., Sankarankuty, A., Tirapelli, D., Catto, J., Agnew, K., Swisher, E., Creaney, J., Robinson, B., Shelley, C.S., Godwin, E.M., Kendall, S., Shipman, C., Bradford, C., Carey, T., Haddad, A., Moyer, J., Peterson, L., Prince, M., Rozek, L., Wolf, G., Bowman, R., Fong, K.M., Yang, I., Korst, R., Rathmell, W.K., Fantacone-Campbell, J.L., Hooke, J.A., Kovatich, A.J., Shriver,

C.D., DiPersio, J., Drake, B., Govindan, R., Heath, S., Ley, T., Tine, B.V., Westervelt, P., Rubin, M.A., Lee, J.I., Aredes, N.D., Mariamidze, A.: Machine learning identifies stemness features associated with oncogenic dedifferentiation. Cell **173**(2), 338–354.e15 (Apr 2018). https://doi.org/10.1016/j.cell.2018.03.034, https://doi.org/10.1016/j.cell.2018.03.034

89. Marx, V.: Method of the year: spatially resolved transcriptomics. Nature methods **18**(1), 9–14 (2021)
90. Merritt, C.R., Ong, G.T., Church, S.E., Barker, K., Danaher, P., Geiss, G., Hoang, M., Jung, J., Liang, Y., McKay-Fleisch, J., et al.: Multiplex digital spatial profiling of proteins and rna in fixed tissue. Nature biotechnology **38**(5), 586–599 (2020)
91. Mimitou, E.P., Lareau, C.A., Chen, K.Y., Zorzetto-Fernandes, A.L., Hao, Y., Takeshima, Y., Luo, W., Huang, T.S., Yeung, B.Z., Papalexi, E., et al.: Scalable, multimodal profiling of chromatin accessibility, gene expression and protein levels in single cells. Nature biotechnology **39**(10), 1246–1258 (2021)
92. Minoura, K., Abe, K., Nam, H., Nishikawa, H., Shimamura, T.: A mixture-of-experts deep generative model for integrated analysis of single-cell multiomics data. Cell reports methods **1**(5), 100071 (2021)
93. Moffitt, J., Zhuang, X.: Chapter one - RNA imaging with multiplexed error-robust fluorescence in situ hybridization (MERFISH). In: Filonov, G.S., Jaffrey, S.R. (eds.) Visualizing RNA dynamics in the cell, Methods in enzymology, vol. 572, pp. 1–49. Academic Press (2016). https://doi.org/https://doi.org/10.1016/bs.mie.2016.03.020, https://www.sciencedirect.com/science/article/pii/S0076687916001324, iSSN: 0076-6879
94. Moncada, R., Barkley, D., Wagner, F., Chiodin, M., Devlin, J.C., Baron, M., Hajdu, C.H., Simeone, D.M., Yanai, I.: Integrating microarray-based spatial transcriptomics and single-cell rna-seq reveals tissue architecture in pancreatic ductal adenocarcinomas. Nature Biotechnology **38** (2020). https://doi.org/https://doi.org/10.1038/s41587-019-0392-8, Dataset Link: https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE111672
95. Monckton, D.G., Jeffreys, A.J.: Dna profiling. Current Opinion in Biotechnology **4**(6), 660–664 (1993)
96. Moor, A.E., Itzkovitz, S.: Spatial transcriptomics: paving the way for tissue-level systems biology. Current opinion in biotechnology **46**, 126–133 (2017)
97. Nestorowa, S., Hamey, F.K., Sala, B.P., Diamanti, E., Shepherd, M., Laurenti, E., Wilson, N.K., Kent, D.G., Göttgens, B.: A single-cell resolution map of mouse hematopoietic stem and progenitor cell differentiation. Blood **128**(8), e20–e31 (Aug 2016). https://doi.org/10.1182/blood-2016-05-716480, https://doi.org/10.1182/blood-2016-05-716480
98. Nettleton, D.F.: Data mining of social networks represented as graphs. Computer Science Review **7**, 1–34 (2013)
99. Nguyen, H., Tran, D., Tran, B., Pehlivan, B., Nguyen, T.: A comprehensive survey of regulatory network inference methods using single cell rna sequencing data. Briefings in bioinformatics **22**(3), bbaa190 (2021)
100. Nguyen, Q.H., Pervolarakis, N., Nee, K., Kessenbrock, K.: Experimental Considerations for Single-Cell RNA Sequencing Approaches **6**, 108. https://doi.org/10.3389/fcell.2018.00108, https://www.frontiersin.org/article/10.3389/fcell.2018.00108/full
101. Palla, G., Spitzer, H., Klein, M., Fischer, D., Schaar, A.C., Kuemmerle, L.B., Rybakov, S., Ibarra, I.L., Holmberg, O., Virshup, I., et al.: Squidpy: a scalable framework for spatial omics analysis. Nature methods **19**(2), 171–178 (2022)

102. Panneerchelvam, S., Norazmi, M.: Forensic dna profiling and database. The Malaysian journal of medical sciences: MJMS **10**(2), 20 (2003)
103. Papalexi, E., Satija, R.: Single-cell RNA sequencing to explore immune cell heterogeneity. Nature Reviews Immunology **18**(1), 35–45 (Aug 2017). https://doi.org/10.1038/nri.2017.76, `https://doi.org/10.1038/nri.2017.76`
104. Pardo, B., Spangler, A., Weber, L.M., Page, S.C., Hicks, S.C., Jaffe, A.E., Martinowich, K., Maynard, K.R., Collado-Torres, L.: spatiallibd: an r/bioconductor package to visualize spatially-resolved transcriptomics data. BMC Genomics **23**(1), 1–5 (2022), Dataset Link: http://research.libd.org/spatialLIBD/
105. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems **32** (2019)
106. Pavlopoulos, G.A., Secrier, M., Moschopoulos, C.N., Soldatos, T.G., Kossida, S., Aerts, J., Schneider, R., Bagos, P.G.: Using graph theory to analyze biological networks. BioData mining **4**(1), 1–27 (2011)
107. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 701–710 (2014)
108. Peterson, V.M., Zhang, K.X., Kumar, N., Wong, J., Li, L., Wilson, D.C., Moore, R., McClanahan, T.K., Sadekova, S., Klappenbach, J.A.: Multiplexed quantification of proteins and transcripts in single cells. Nature biotechnology **35**(10), 936–939 (2017)
109. Pham, D., Tan, X., Xu, J., Grice, L.F., Lam, P.Y., Raghubar, A., Vukovic, J., Ruitenberg, M.J., Nguyen, Q.: stlearn: integrating spatial location, tissue morphology and gene expression to find cell types, cell-cell interactions and spatial trajectories within undissociated tissues. BioRxiv (2020)
110. Picelli, S.: Single-cell RNA-sequencing: The future of genome biology is now **14**(5), 637–650. https://doi.org/10.1080/15476286.2016.1201618
111. Pouyanfar, S., Sadiq, S., Yan, Y., Tian, H., Tao, Y., Reyes, M.P., Shyu, M.L., Chen, S.C., Iyengar, S.S.: A survey on deep learning: Algorithms, techniques, and applications. ACM Computing Surveys (CSUR) **51**(5), 1–36 (2018)
112. Qi, F., Qian, S., Zhang, S., Zhang, Z.: Single cell RNA sequencing of 13 human tissues identify cell types and receptors of human coronaviruses. Biochemical and Biophysical Research Communications **526**(1), 135–140 (May 2020). https://doi.org/10.1016/j.bbrc.2020.03.044, `https://doi.org/10.1016/j.bbrc.2020.03.044`
113. Qiu, P.: Embracing the dropouts in single-cell RNA-seq analysis. Nature Communications **11**(1), 1169 (Dec 2020). https://doi.org/10.1038/s41467-020-14976-9, `http://www.nature.com/articles/s41467-020-14976-9`
114. Rao, J., Zhou, X., Lu, Y., Zhao, H., Yang, Y.: Imputing single-cell rna-seq data by combining graph convolution and autoencoder neural networks. Iscience **24**(5), 102393 (2021)
115. Rizvi, A.H., Camara, P.G., Kandror, E.K., Roberts, T.J., Schieren, I., Maniatis, T., Rabadan, R.: Single-cell topological RNA-seq analysis reveals insights into cellular differentiation and development. Nature Biotechnology **35**(6), 551–560 (May 2017). https://doi.org/10.1038/nbt.3854, `https://doi.org/10.1038/nbt.3854`
116. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning internal representations by error propagation. Tech. rep., California Univ San Diego La Jolla Inst for Cognitive Science (1985)

117. Ruoff, F., Henes, M., Templin, M., Enderle, M., Bösmüller, H., Wallwiener, D., Brucker, S.Y., Schenke-Layland, K., Weiss, M.: Targeted protein profiling of in vivo nipp-treated tissues using digiwest technology. Applied Sciences **11**(23), 11238 (2021)
118. Shao, X., Yang, H., Zhuang, X., Liao, J., Yang, P., Cheng, J., Lu, X., Chen, H., Fan, X.: scdeepsort: a pre-trained cell-type annotation method for single-cell transcriptomics using deep learning with a weighted graph neural network. Nucleic acids research **49**(21), e122–e122 (2021)
119. Shi, Y., Paige, B., Torr, P., et al.: Variational mixture-of-experts autoencoders for multi-modal deep generative models. Advances in Neural Information Processing Systems **32** (2019)
120. Song, Q., Su, J.: Dstg: deconvoluting spatial transcriptomics data through graph-based artificial intelligence. Briefings in Bioinformatics **22**(3), 1–13 (2021)
121. Song, Q., Su, J., Zhang, W.: scgcn is a graph convolutional networks algorithm for knowledge transfer in single cell omics. Nature Communications **12**(1), 1–11 (2021)
122. Stanley III, J.S., Gigante, S., Wolf, G., Krishnaswamy, S.: Harmonic alignment. In: Proceedings of the 2020 SIAM International Conference on Data Mining. pp. 316–324. SIAM (2020)
123. Stickels, R.R., Murray, Evan Kumar, P., Li, J., Marshall, J.L., Di Bella, D.J., Arlotta, P., Macosko, E.Z., Chen, F.: Highly sensitive spatial transcriptomics at near-cellular resolution with slide-seqv2. Nature Biotechnology (03 2021). https://doi.org/10.1038/s41587-020-0739-1
124. Stoeckius, M., Hafemeister, C., Stephenson, W., Houck-Loomis, B., Chattopadhyay, P.K., Swerdlow, H., Satija, R., Smibert, P.: Simultaneous epitope and transcriptome measurement in single cells. Nature methods **14**(9), 865–868 (2017)
125. Stuart, T., Butler, A., Hoffman, P., Hafemeister, C., Papalexi, E., Mauck III, W.M., Hao, Y., Stoeckius, M., Smibert, P., Satija, R.: Comprehensive integration of single-cell data. Cell **177**(7), 1888–1902 (2019)
126. Ståhl, P.L., Salmén, F., Vickovic, S., Lundmark, A., Navarro, J.F., Magnusson, J., Giacomello, S., Asp, M., Westholm, J.O., Huss, M., Mollbrink, A., Linnarsson, S., Codeluppi, S., Åke Borg, Pontén, F., Costea, P.I., Sahlén, P., Mulder, J., Bergmann, O., Lundeberg, J., Frisén, J.: Visualization and analysis of gene expression in tissue sections by spatial transcriptomics. Science **353**(6294), 78–82 (2016). https://doi.org/10.1126/science.aaf2403
127. Subramanian, A., Tamayo, P., Mootha, V.K., Mukherjee, S., Ebert, B.L., Gillette, M.A., Paulovich, A., Pomeroy, S.L., Golub, T.R., Lander, E.S., et al.: Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. Proceedings of the National Academy of Sciences **102**(43), 15545–15550 (2005)
128. Tan, Y., Cahan, P.: Singlecellnet: a computational tool to classify single cell rna-seq data across platforms and across species. Cell systems **9**(2), 207–213 (2019)
129. Tang, F., Barbacioru, C., Wang, Y., Nordman, E., Lee, C., Xu, N., Wang, X., Bodeau, J., Tuch, B.B., Siddiqui, A., et al.: mrna-seq whole-transcriptome analysis of a single cell. Nature methods **6**(5), 377–382 (2009)
130. Teves, J.M., Won, K.J.: Mapping cellular coordinates through advances in spatial transcriptomics technology. Molecules and Cells **43**(7), 591 (2020)
131. Tian, T., Wan, J., Song, Q., Wei, Z.: Clustering single-cell rna-seq data with a model-based deep learning approach. Nature Machine Intelligence **1**(4), 191–198 (2019)

132. Tian, T., Zhang, J., Lin, X., Wei, Z., Hakonarson, H.: Model-based deep embedding for constrained clustering analysis of single cell rna-seq data. Nature communications **12**(1), 1–12 (2021)
133. Trapnell, C.: Defining cell types and states with single-cell genomics. Genome Research **25**(10), 1491–1498 (Oct 2015). https://doi.org/10.1101/gr.190595.115, https://doi.org/10.1101/gr.190595.115
134. Travaglini, K.J., Nabhan, A.N., Penland, L., Sinha, R., Gillich, A., Sit, R.V., Chang, S., Conley, S.D., Mori, Y., Seita, J., Berry, G.J., Shrager, J.B., Metzger, R.J., Kuo, C.S., Neff, N., Weissman, I.L., Quake, S.R., Krasnow, M.A.: A molecular cell atlas of the human lung from single-cell RNA sequencing. Nature **587**(7835), 619–625 (Nov 2020). https://doi.org/10.1038/s41586-020-2922-4, https://doi.org/10.1038/s41586-020-2922-4
135. Valdes-Mora, F., Handler, K., Law, A.M., Salomon, R., Oakes, S.R., Ormandy, C.J., Gallego-Ortega, D.: Single-cell transcriptomics in cancer immunobiology: the future of precision oncology. Frontiers in Immunology **9**, 2582 (2018)
136. Van Valen, D.A., Kudo, T., Lane, K.M., Macklin, D.N., Quach, N.T., DeFelice, M.M., Maayan, I., Tanouchi, Y., Ashley, E.A., Covert, M.W.: Deep learning automates the quantitative analysis of individual cells in live-cell imaging experiments. PLoS computational biology **12**(11), e1005177 (2016)
137. Villani, A.C., Satija, R., Reynolds, G., Sarkizova, S., Shekhar, K., Fletcher, J., Griesbeck, M., Butler, A., Zheng, S., Lazo, S., Jardine, L., Dixon, D., Stephenson, E., Nilsson, E., Grundberg, I., McDonald, D., Filby, A., Li, W., Jager, P.L.D., Rozenblatt-Rosen, O., Lane, A.A., Haniffa, M., Regev, A., Hacohen, N.: Single-cell RNA-seq reveals new types of human blood dendritic cells, monocytes, and progenitors. Science **356**(6335) (Apr 2017). https://doi.org/10.1126/science.aah4573, https://doi.org/10.1126/science.aah4573
138. Wang, G., Moffitt, J.R., Zhuang, X.: Multiplexed imaging of high-density libraries of rnas with merfish and expansion microscopy. Scientific reports **8**(1), 1–13 (2018)
139. Wang, J., Ma, A., Chang, Y., Gong, J., Jiang, Y., Qi, R., Wang, C., Fu, H., Ma, Q., Xu, D.: scGNN is a novel graph neural network framework for single-cell RNA-Seq analyses **12**(1), 1882
140. Wang, J., Ma, A., Chang, Y., Gong, J., Jiang, Y., Qi, R., Wang, C., Fu, H., Ma, Q., Xu, D.: scgnn is a novel graph neural network framework for single-cell rna-seq analyses. Nature communications **12**(1), 1–11 (2021)
141. Wang, M., Zheng, D., Ye, Z., Gan, Q., Li, M., Song, X., Zhou, J., Ma, C., Yu, L., Gai, Y., et al.: Deep graph library: A graph-centric, highly-performant package for graph neural networks. arXiv preprint arXiv:1909.01315 (2019)
142. Waylen, L.N., Nim, H.T., Martelotto, L.G., Ramialison, M.: From whole-mount to single-cell spatial assessment of gene expression in 3d. Communications biology **3**(1), 1–11 (2020)
143. Wen, H., Ding, J., Jin, W., Wang, Y., Xie, Y., Tang, J.: Graph neural networks for multimodal single-cell data integration. In: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. pp. 4153–4163 (2022)
144. Wolf, F.A., Angerer, P., Theis, F.J.: Scanpy: large-scale single-cell gene expression data analysis. Genome biology **19**(1), 1–5 (2018)
145. Wu, J., Xiao, Y., Sun, J., Sun, H., Chen, H., Zhu, Y., Fu, H., Yu, C., E., W., Lai, S., Ma, L., Li, J., Fei, L., Jiang, M., Wang, J., Ye, F., Wang, R., Zhou, Z., Zhang, G., Zhang, T., Ding, Q., Wang, Z., Hao, S., Liu, L., Zheng, W., He, J., Huang, W., Wang, Y., Xie, J., Li, T., Cheng, T., Han, X., Huang, H., Guo, G.: A single-cell survey of cellular hierarchy in acute myeloid leukemia. Journal of Hematology

&amp Oncology **13**(1) (Sep 2020). https://doi.org/10.1186/s13045-020-00941-y, https://doi.org/10.1186/s13045-020-00941-y

146. Wu, K.E., Yost, K.E., Chang, H.Y., Zou, J.: Babel enables cross-modality translation between multiomic profiles at single-cell resolution. Proceedings of the National Academy of Sciences **118**(15), e2023070118 (2021)
147. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Philip, S.Y.: A comprehensive survey on graph neural networks. IEEE transactions on neural networks and learning systems **32**(1), 4–24 (2020)
148. Xie, S., Yu, Z., Lv, Z.: Multi-disease prediction based on deep learning: a survey. CMES-computer Modeling in Engineering and Sciences (2021)
149. Yang, K.D., Belyaeva, A., Venkatachalapathy, S., Damodaran, K., Katcoff, A., Radhakrishnan, A., Shivashankar, G., Uhler, C.: Multi-domain translation between single-cell imaging and sequencing data using autoencoders. Nature communications **12**(1), 1–10 (2021)
150. Yeung, K.Y., Ruzzo, W.L.: Details of the adjusted rand index and clustering algorithms, supplement to the paper an empirical study on principal component analysis for clustering gene expression data. Bioinformatics **17**(9), 763–774 (2001)
151. Yu, Z., Lu, Y., Wang, Y., Tang, F., Wong, K.C., Li, X.: Zinb-based graph embedding autoencoder for single-cell rna-seq interpretations. Proceedings of the AAAI Conference on Artificial Intelligence **36**(4), 4671–4679 (2022)
152. Zeng, P., Lin, Z.: Couple coc+: an information-theoretic co-clustering-based transfer learning framework for the integrative analysis of single-cell genomic data. PLoS Computational Biology **17**(6), e1009064 (2021)
153. Zeng, W., Chen, X., Duren, Z., Wang, Y., Jiang, R., Wong, W.H.: Dc3 is a method for deconvolution and coupled clustering from bulk and single-cell genomics data. Nature communications **10**(1), 1–11 (2019)
154. Zhang, J.Y., Wang, X.M., Xing, X., Xu, Z., Zhang, C., Song, J.W., Fan, X., Xia, P., Fu, J.L., Wang, S.Y., Xu, R.N., Dai, X.P., Shi, L., Huang, L., Jiang, T.J., Shi, M., Zhang, Y., Zumla, A., Maeurer, M., Bai, F., Wang, F.S.: Single-cell landscape of immunological responses in patients with COVID-19. Nature Immunology **21**(9), 1107–1118 (Aug 2020). https://doi.org/10.1038/s41590-020-0762-x, https://doi.org/10.1038/s41590-020-0762-x
155. Zheng, G.X.Y., Terry, J.M., Belgrader, P., Ryvkin, P., Bent, Z.W., Wilson, R., Ziraldo, S.B., Wheeler, T.D., McDermott, G.P., Zhu, J., Gregory, M.T., Shuga, J., Montesclaros, L., Underwood, J.G., Masquelier, D.A., Nishimura, S.Y., Schnall-Levin, M., Wyatt, P.W., Hindson, C.M., Bharadwaj, R., Wong, A., Ness, K.D., Beppu, L.W., Deeg, H.J., McFarland, C., Loeb, K.R., Valente, W.J., Ericson, N.G., Stevens, E.A., Radich, J.P., Mikkelsen, T.S., Hindson, B.J., Bielas, J.H.: Massively parallel digital transcriptional profiling of single cells **8**(1), 14049. https://doi.org/10.1038/ncomms14049, http://www.nature.com/articles/ncomms14049, Dataset Link: https://support.10xgenomics.com/single-cell-gene-expression/datasets/3.1.0/5k$_pbmc_protein_v3$
156. Zheng, G.X., Terry, J.M., Belgrader, P., Ryvkin, P., Bent, Z.W., Wilson, R., Ziraldo, S.B., Wheeler, T.D., McDermott, G.P., Zhu, J., et al.: Massively parallel digital transcriptional profiling of single cells. Nature communications **8**(1), 1–12 (2017), Dataset Link: https://support.10xgenomics.com/single-cell-gene-expression/datasets/2.1.0/pbmc4k
157. Zuo, C., Chen, L.: Deep-joint-learning analysis model of single cell transcriptome and open chromatin accessibility data. Briefings in Bioinformatics **22**(4), bbaa287 (2021)

158. Zuo, C., Dai, H., Chen, L.: Deep cross-omics cycle attention model for joint analysis of single-cell multi-omics data. Bioinformatics **37**(22), 4091–4099 (2021)

A Environment Dependencies

For the dependencies of our first version of the DANCE package, please refer to Table 3 for more details.

Dependency	Version
h5py	$\geq$3.7.0
leidenalg	$\geq$0.8.10
networkx	$\geq$2.8.5
numba	$\geq$0.56.0
opencv-python	$\geq$4.6.0.66
openpyxl	$\geq$3.0.10
psutil	$\geq$5.9.1
pyro-ppl	$\geq$1.8.1
python-igraph	$\geq$0.9.11
rdata	$\geq$0.8
scanpy	$\geq$1.9.1
scikit-learn	$\geq$1.1.2
scikit-misc	$\geq$0.1.4
scipy	$\geq$1.9.0
seaborn	$\geq$0.11.2
skorch	$\geq$0.11.0
statsmodels	$\geq$0.13.2
torch	$\geq$1.11.0
torchnmf	$\geq$0.3.4
torchvision	$\geq$0.12.0
tqdm	$\geq$4.64.0

Table 3: Dependencies of DANCE Package

B Performance Display

In this section, we show the performance comparison between our implementation and the original implementation for tasks as below:

- **Single Modality Module:** Imputation, Cell Type Annotation and Clustering.
- **Multi-modality Module:** Modality Prediction, Modality Matching and Joint Embedding
- **Spatial Module:** Spatial Domain and Cell Type Deconvolution.

If the original implementation is unavailable to the public or is based on non-python language, we re-implement them with generic python language in our DANCE package. If the original implementation exists, we transform them into the generic fit-predict-score structure. Note that not all methods are tested on all standard benchmark datasets. To bridge this gap, we test them on each standard benchmark dataset for system evaluation and fair comparison. In all performance comparison tables, we compare the performance of our re-implemented algorithms denoted as "current" with that of original ones denoted as "reported". Note that N/A means that the performance is not available on this dataset.

Table 4: Imputation Task: performance comparison between our and original implementations in DANCE.

Model	Evaluation Metric	Mouse Brain (current/reported)	Mouse Embryo (current/reported)
DeepImpute	MSE	**0.12** / N/A	**0.12** / N/A
ScGNN	MSE	0.47 / N/A	1.10 / N/A
GraphSCI	MSE	0.42 / N/A	0.87 / N/A
Louvain	ARI	0.31 / 0.33	0.2528 / N/A

Table 5: Cell Type Annotation: performance comparison between our and original implementations in DANCE. (Note: * Benchmark datasets have been renormalized when running the original implementation of Celltypist to meet its format requirements.)

Model	Evaluation Metric	Mouse Brain 2695 (current/reported)	Mouse Spleen 1759 (current/reported)	Mouse Kidney 203 (current/reported)
scDeepsort	ACC	0.363/0.363	0.965 /0.965	**0.901**/0.911
Celltypist*	ACC	0.680/0.666	0.966/0.848	0.879/0.832
singleCellNet	ACC	0.693/0.803	**0.975**/0.975	0.795/0.842
ACTINN	ACC	**0.860**/0.778	0.516/0.236	0.829/0.798
SVM	ACC	0.683/0.683	0.056/0.049	0.704/0.695

Table 6: Clustering Task: performance comparison between our and original implementations in DANCE.

Model	Evaluation Metric	10x PBMC (current/reported)	Mouse ES (current/reported)	Worm Neuron (current/reported)	Mouse Bladder (current/reported)
graph-sc	ARI	0.72 / 0.70	0.82 / 0.78	**0.57** / 0.46	**0.68** / 0.63
scDCC	ARI	**0.82** / 0.81	**0.98** / N/A	0.51 / 0.58	0.60 / 0.66
scDeepCluster	ARI	0.81 / 0.78	0.98 / 0.97	0.51 / 0.52	0.56 / 0.58
scDSC	ARI	0.72 / 0.78	0.84 / N/A	0.46 / 0.65	0.65 / 0.72
scTAG	ARI	0.75 / N/A	0.96 / N/A	0.53 / N/A	0.60 / N/A

Table 7: Modality Prediction Task: performance comparison between our and original implementations in DANCE.

Model	Evaluation Metric	GEX2ADT (current/reported)	ADT2GEX (current/reported)	GEX2ATAC (current/reported)	ATAC2GEX (current/reported)
ScMoGCN	RMSE	**0.3885** / 0.3885	**0.3242** / 0.3242	**0.1778** / 0.1778	**0.2315** / 0.2315
SCMM	RMSE	0.6264 / N/A	0.4458 / N/A	0.2163 / N/A	0.3730 / N/A
Cross-modal autoencoders	RMSE	0.5725 / N/A	0.3585 / N/A	0.1917 / N/A	0.2551 / N/A
BABEL	RMSE	0.4335 / N/A	0.3673 / N/A	0.1816 / N/A	0.2394 / N/A
scTAG	ARI	0.75 / N/A	0.96 / N/A	0.53 / N/A	0.60 / N/A

Table 8: Modality Matching Task: performance comparison between our and original implementations in DANCE.

Model	Evaluation Metric	GEX2ADT (current/reported)	GEX2ATAC (current/reported)
ScMoGCN	ACC	**0.0827** / 0.0810	**0.0600** / 0.0630
SCMM	ACC	0.005 / N/A	5e-5 / N/A
Cross-modal autoencoders	ACC	0.0002 / N/A	0.0002 / N/A

Table 9: Joint Embedding Task: performance comparison between our and original implementations in DANCE.

Model	Evaluation Metric	GEX2ADT (current/reported)	GEX2ATAC (current/reported)
ScMoGCN	ARI	0.706 / N/A	0.702 / N/A
ScMoGCNv2	ARI	**0.734** / N/A	N/A / N/A
scMVAE	ARI	0.499 / N/A	0.577 / N/A
scDEC(JAE)	ARI	0.705 / N/A	**0.735** / N/A
DCCA	ARI	0.35 / N/A	0.381 / N/A

Table 10: Spatial Domain Task: performance comparison between our and original implementations in DANCE.

Model	Evaluation Metric	Slice 151673 (current/reported)	Slice 151676 (current/reported)	Slice 151507 (current/reported)
SpaGCN	ARI	0.51 / 0.522	0.41 / N/A	0.45 / N/A
STAGATE	ARI	**0.59** / N/A	**0.60** / 0.60	**0.608** / N/A
stLearn	ARI	0.30 / 0.36	0.29 / N/A	0.31 / N/A
Louvain	ARI	0.31 / 0.33	0.2528 / N/A	0.28 / N/A

Table 11: Cell Type Deconvolution: performance comparison between our and original implementations in DANCE.

Model	Evaluation Metric	GSE174746 (current/reported)	CARD Synthetic (current/reported)	SPOTlight Synthetic (current/reported)
DSTG	MSE	0.172 / N/A	0.0247 / N/A	0.042 / N/A
SpatialDecon	MSE	0.0014 / 0.009	**0.0077** / N/A	**0.0055** / N/A
SPOTlight	MSE	0.0098 / N/A	0.0246 / 0.118	0.0109 / 0.16
CARD	MSE	**0.0012** / N/A	0.0078 / 0.0062	0.0076 / N/A

C Details of Supported Datasets

All supported datasets across 8 tasks in DANCE are summarized in Table 12. For each supported dataset, we list what type of species and tissue it is about, dataset dimensions including the number of cells and genes, and also the protocol about how to generate the dataset for reference. In the column of "Availability", the dataset link is provided once you click the reference.

Table 12: A summary of all supported datasets in DANCE.

Module	Task	Dataset	Species&Tissue	Dataset Dimensions	Protocol	Availability
Single Modality	Imputation	10X PBMC 5K	Human, PBMC	5,247 cells 33,570 genes	10x Genomics	[155]
		Human Embryonic stem cells (ESC)	Human, ESC	758 cells 17,826 genes	Illumina HiSeq 2500	[25]
		Mouse Neuron Cells 10k	Mouse, Neuron	11,843 cells 31,053 genes	10x Genomics	[155]
		Mouse ESC	Mouse, Neuron	2,717 cells 24,175 genes	Droplet Barcoding	[65]
	Cell Type Annotation	HCL	Human	562,977 cells 56 tissues	Smart-seq2	[53]
		MCA	Mouse	201,764 cells 32 tissues	Smart-seq2	[52]
	Clustering	10X PBMC 4K	Human, PBMC	4,271 cells 16,653 genes	10x Genomics	[156]
		Mouse Bladder Cells	Mouse, Bladder	2,746 cells 20,670 genes	Microwell-seq	[51]
		Worm Neuron Cells	Worm, Nerve	4,186 cells 13,488 genes	sci-RNA-seq	[17]
		Mouse Embryonic Stem Cells	Mouse, Embryo	2,717 cells 24,175 genes	Droplet Barcoding	[65]
Multimodality	Modality Prediction	Openproblems Neurips2021 CITE	Human, BMMC	81,241 cells 13,953 genes 134 surface proteins	10X TotalSeq B	[5]
		Openproblems Neurips2021 Multiome	Human, BMMC	62,501 cells 13,431 genes 116,490 peaks	10X Multiome	[5]
	Modality Matching	Openproblems Neurips2021 CITE	Human, BMMC	81,241 cells 13,953 genes 134 surface proteins	10X TotalSeq B	[5]
		Openproblems Neurips2021 Multiome	Human, BMMC	62,501 cells 13,431 genes 116,490 peaks	10X Multiome	[5]
	Joint Embedding	Openproblems Neurips2021 CITE	Human, BMMC	81,241 cells 13,953 genes 134 surface proteins	10X TotalSeq B	[5]
		Openproblems Neurips2021 Multiome	Human, BMMC	62,501 cells 13,431 genes 116,490 peaks	10X Multiome	[5]
Spatial	Spatial Domain	LIBD Human Dorsolateral Prefrontal Cortex	Human, Dorsolateral prefrontal cortex	12 slices Slice 151673: 3,639 spots 33,538 genes	10X Visium	[104]
	Cell Type Deconvolution	Mouse Posterior Brain	Mouse, Posterior brain	3,353 spots 31,053 genes	10X Visium	[2]
		Mouse Olfactory Bulb	Mouse, Olfactory bulb	1,185 spots 11,176 genes	10X Visium	[1]
		HEK293T and CCRF-CEM	Human	56 mixtures 1,414 genes	NanoString GeoMx	[29]
		Human PDAC	Human, Pancreas	3,353 spots 31,053 genes	Spatial Transcriptomics	[94]

D Details of Supported Tasks and Models

D.1 Single Modality Module

Imputation The goal of imputation for scRNA-seq data is to address artificial zeros in scRNA-seq data generated during the sequencing process systematically or by chance due to technological limitations. Imputation aims at correcting these artificial zeros by filling in realistic values that reflect true biological gene expressions [71]. Thus, a good imputation method should be able to distinguish artificial zeros from biologically true zeros and recover true expressions for artificial zeros. As the corresponding biologically true expression values are unavailable for entries of artificial zeros in the gene-cell matrix, dropouts are simulated for benchmarking such that metrics such as cosine similarity, correlations, or MSE-related metrics can then be used to evaluate imputation algorithms; alternatively, prior cell information, such as cell types, can be used to evaluate whether an imputation method could recover biological signals by comparing clustering patterns using original and imputed expressions [6,31,56,75,114,139].

dance.modules.single_modality.imputation.deepimpute DeepImpute [6] builds multiple neural networks in parallel to impute target genes using a set of input genes. Given a scRNA-seq matrix X, target genes, i.e, genes to be imputed, are selected based on the variance over mean ratio, which are split into N random subsets. Each subset corresponds to a neural network consisting of two layers: a dense layer and a dropout layer with ReLu and softplus as activations, respectively. The inputs to each neural network are genes highly correlated with corresponding target genes based on Pearson's correlation coefficient. The loss function is the Weighted mean squared error (MSE).

dance.modules.single_modality.imputation.scgnn scGNN [139] uses an integrative autoencoder framework for scRNA-seq gene expression imputation that incorporates gene regulatory signals (TRS). It includes a feature autoencoder, a graph autoencoder, and a cluster autoencoder that are trained iteratively, whose outputs are used in the final imputation autoencoder to recover gene expressions.

scgnn uses left-truncated mixed Gaussian (LTMG) to account for regulatory signals. The normalized expression for a gene is modeled by a mixture of k Gaussian distributions representing k TRSs, and left truncation is used to account for dropouts and lowly expressed values. The expression of gene i in cell j can then be assigned to the TRS under whose Gaussian distribution the observed value is most likely to occur. All parameters in this step are estimated by MLE.

Given an input scRNA-seq expression matrix X, the feature autoencoder extracts a lower-dimensional embedding X' and reconstructs the expression $\hat{X}$, which is composed of two dense layers in both the encoder and the decoder. The loss function is MSE integrated with gene regulation information

$$\mathcal{L} = (1-\alpha)\sum(X-\hat{X})^2 + \alpha\sum\left((X-\hat{X})^2 \odot \mathrm{TRS}\right) \tag{1}$$

where $\alpha \in [0, 1]$ and $\odot$ denotes element-wise product.

A KNN cell graph is constructed from the learned embeddings in the feature autoencoder, which is pruned by the Isolation Forest. With node feature matrix X', the two-layer GCN encoder is constructed as

$$Z = \mathrm{ReLU}(\tilde{A}\mathrm{ReLU}(\tilde{A}X'W_1)W_2). \tag{2}$$

where $\tilde{A}$ is the symmetrically normalized adjacency matrix of the cell graph and W_1, W_2 are learnable parameters.

The associated decoder is:

$$\hat{A} = \mathrm{sigmoid}(ZZ^\top) \tag{3}$$

The parameters are learned through the cross-entropy loss

$$\mathcal{L}(A, \hat{A}) = -\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}(a_{ij} * log(\hat{a}_{ij}) + (1 - a_{ij}) * \log(1 - \hat{a}_{ij})) \tag{4}$$

where a_{ij} and $\hat{a}_{ij}$ are elements in A and $\hat{A}$.

The k-means clustering is then applied to the learned embedding in the graph autoencoder. The number of clusters is determined by the Louvain algorithm. Each cell cluster has an autoencoder to regenerate gene expressions within a cluster, whose structure is the same as the feature autoencoder without being regularized by TRS. The reconstructed gene expression is fed back into the feature encoder iteratively until the convergence of adjacency matrix ($\tilde{A}_t - \tilde{A}_{t-1} < \gamma_1$)and cell clusters converge (ARI$> \gamma_2$).

After convergence, a final imputation autoencoder is trained, which has a similar structure and loss to the feature autoencoder but with three additional regularizations:

$$\begin{aligned}\mathcal{L} = (1-\alpha)\sum(X - \hat{X})^2 + \alpha\sum\left((X - \hat{X})^2 \circ \mathrm{TRS}\right) \\ + \beta\sum|w| + \gamma_1\sum\left(A \cdot (X - \hat{X})^2\right) + \gamma_2\sum\left(B \cdot (X - \hat{X})^2\right)\end{aligned} \tag{5}$$

where $\beta \in [0, 1]$ controls the intensity of the $L1$ penalization and B is a matrix indicating whether two cells belong to the same cluster.

dance.modules.single_modality.imputation.graphsci GraphSCI [114] is a GNN-based method to impute scRNA-seq data expressions. Given a gene expression matrix X with N genes and M cells, a gene graph associated with an $N \times N$ adjacency matrix A is derived based on gene-wise Pearson correlation coefficients. The gene expression matrix and the constructed gene graph are the input to 1) a two-layer GCN whose lower-dimensional embedding gives the Gaussian distribution representing gene relationships and 2) a two-layer fully

connected neural network (NN) from which a ZINB distribution for scRNA-seq data expressions can be inferred. In particular, the GCN is formulated as

$$H_{\mathcal{N}}^{(1)} = \mathrm{ReLU}(\tilde{A}XW_{\mathcal{N}}^{(0)}) \tag{6}$$

$$[\mu_{\mathcal{N}}, \sigma_{\mathcal{N}}^2] = \tilde{A}H_{\mathcal{N}}^{(1)}W_{\mathcal{N}}^{(1)} \tag{7}$$

where $W_{\mathcal{N}}^{(0)}$, $W_{\mathcal{N}}^{(1)}$ are the trainable parameters.

The fully-connected neural network is defined as

$$H_{\mathcal{M}}^{(1)} = \tanh(X^\top(W_{\mathcal{M}}^{(0)} \odot A) + b^{(0)}) \tag{8}$$

$$[\mu_{\mathcal{M}}, \theta_{\mathcal{M}}, \pi_{\mathcal{M}}] = \sigma(H_{\mathcal{M}}^{(1)}W_{\mathcal{M}}^{(1)} + b^{(1)}) \tag{9}$$

where $\mu_{\mathcal{M}}, \theta_{\mathcal{M}}, \pi_{\mathcal{M}}$ are the mean, dispersion, and dropout probability for the ZINB distribution, and $W_{\mathcal{M}}^{(0)}$, $W_{\mathcal{M}}^{(1)}, b^{(0)}, b^{(1)}$ are trainable parameters.

The learned latent variables $([\mu_{\mathcal{N}}, \sigma_{\mathcal{N}}^2], [\mu_{\mathcal{M}}, \theta_{\mathcal{M}}, \pi_{\mathcal{M}}]$ are re-parameterized as $Z^{\mathcal{N}}$ and $Z^{\mathcal{M}}$, i.e, the latent space representation of the GCN and NN, to be used by the decoder to construct the imputed gene expression $\hat{X}$. Specifically, the imputed gene expression for gene i in cell j is

$$\hat{X}_{ij} = g_{\varphi_1}(Z^{cM_j}) = diag(\vec{s}_j) \times Z_j^{\mathcal{M}} \tag{10}$$

where $\vec{s}_j$ is the size factor of cell j.

Meanwhile, the reconstructed edge weight between gene i and gene j is

$$\hat{A}_{ij} = g_{\varphi_2}(Z_i^{\mathcal{N}}, Z_j^{\mathcal{N}}) = \mathrm{sigmoid}((Z_i^{\mathcal{N}})^\top Z_j^{\mathcal{N}}) \tag{11}$$

$Z^{\mathcal{N}}$ and $Z^{\mathcal{M}}$ are optimized by variational lower bound

$$\begin{aligned}\mathcal{L}(\phi, \varphi) = \mathbb{E}_{q_\phi}\left[\sum_{i\in\mathcal{N}, j\in\mathcal{M}} \log p_{\varphi_1}(\hat{X}_{ij}|Z_i^{\mathcal{N}}, Z_j^{\mathcal{M}})\right] + \mathbb{E}_{q_\phi}\left[\sum_{i,j\in\mathcal{N}} \log p_{\varphi_2}(\hat{A}_{ij}|Z_i^{\mathcal{N}}, Z_j^{\mathcal{N}})\right] \\ - D_{KL}(q_\phi(Z^{\mathcal{M}}|A, X^\top)||p(Z^{\mathcal{M}})) - D_{KL}(q_\phi(Z^{\mathcal{N}}|A, X^\top)||p(Z^{\mathcal{N}}))\end{aligned} \tag{12}$$

where $\mathbb{E}_{q_\phi}$ is the cross-entropy function and and $D_{KL}(q||p)$ is the Kullback-Leibler divergence between distributions q and p.

Cell Type Annotation Cell type annotation targets applying statistics of cellular properties to infer cell types. Given the gene expression of several cell types, for each cell with a certain single-cell expression matrix, the degree of similarity can be calculated. Based on the optimal similarity result, the cell type can then be inferred. In DANCE, we support 5 models that establish measurements of evaluating the similarity of gene expression profiles of unknown cells to gene

expression matrices of known cell types. The model performance is evaluated by prediction accuracy.

dance.modules.single_modality.cell_type_annotation.scdeepsort Scdeepsort [118] includes three modules: an embedding layer, a weighted graph aggregator, and linear classification towers. Given m genes and n cells, we have the input single-cell data matrix $D \in \mathbb{R}^{m \times n}$. To generate the weighted cell-gene graph, we first apply PCA to extract d dimensional representations of initial representations. A weighted adjacency matrix $A \in \mathbb{R}^{(m+n)^2}$ and a node embedding $X \in \mathbb{R}^{(m+n) \times d}$ are generated from D. For a gene node j, the shareable parameter β_j denotes the confidence value for the edges interacting with node j. For the self-loop edge for each cell, we use α to denote its confidence value. Let h_i^k be the node i's embedding vector in the k_{th} layer, the aggregator layer is

$$h_i^k = \sigma(W^{k-1} \frac{\alpha h_i^{k-1} + \sum_{j \in N(i)} \beta_j a_{ij} h_j^{k-1}}{1 + |N(i)|} + b^{k-1}), \tag{13}$$

where $a_{ij}$is the normalized weight of an edge from nodes i to j. Then cell node representations are fed into linear classifier layers

$$\hat{y}_i = softmax(W h_i^k + b) \tag{14}$$

Cross entropy loss is applied in the objective function:

$$\hat{\theta} = \text{argmin}_\theta \sum_{c=1}^{C} y_c log \hat{y}_c, \tag{15}$$

where θ represents all trainable parameters.
dance.modules.single_modality.cell_type_annotation.celltypist Celltypist [32] uses a multinomial logistic regression classifier defined as:

$$l(x) = \frac{1}{1 + e^{-w^t x}}, \tag{16}$$

where $l(x)$ is the decision score with x as the input vector. Then the decision score for each cell is defined as the linear combination of the scaled gene expression and the model coefficients associated with the given gene type, and the possibility is calculated by transforming the decision score by a sigmoid function.
dance.modules.single_modality.cell_type_annotation.singlecellnet SingleCellnet [128] revamped the random forest classifier method to enable classification of scRNA-seq data cross platforms and cross-species. It sends the input features into a number of decision tree classifiers and uses majority voting to make predictions.
dance.modules.single_modality.cell_type_annotation.actinn ACTINN [81] proposes a neural network based model for cell type annotation. It applies multilayer perceptron for the identification of cell types that can be implemented as:

$$x_i = g(W_i x_{i-1} + b_{i-1}), \tag{17}$$

where x_i is the output from the i_{th} layer, W_i and b_i represent the weight matrix and the bias in the i_th layer and g represents the activation function used in the neural network. In the input layers and hidden layers, the activation function is ReLU as:

$$ReLU(x) = max(0, x). \tag{18}$$

It utilizes the softmax function as the activation function g for the output layer, which is defined as:

$$softmax(x_j) = \frac{exp(x_j)}{\sum_{i=1}^{k} exp(x_i)} \tag{19}$$

where x_j represents the j_{th} element of the input vector for the output layer and k is the length of the vector x.

dance.modules.single_modality.cell_type_annotation.svm Support vector machine(SVM) is widely adopted as a benchmark in many studies [3,118]. Given the input x and label y, the prediction function is

$$\hat{y} = wx + b, \tag{20}$$

where w represents the weights and b is the interception in the SVM. Based on Eq 20, SVM optimizes the following problem:

$$\min_{w,b} \frac{1}{2}||w||^2,$$

$$subject\ to\ y(wx + b) - 1 \geq 0.$$

Clustering Clustering is a crucial part of single-cell analysis. With clustering, researchers can identify cell types or cell-type subgroups within the gene expression data. In the clustering task, we now support 5 models. The first 3 models are GNN based, and the later 2 models are non-GNN based with autoencoder as the backbone. The clustering performance is evaluated by Adjusted Rand index (ARI).

dance.modules.single_modality.clustering.scdeepcluster scDeepCluster [131] introduces a ZINB-based autoencoder. The input matrix is corrupted by a Gaussian noise e: $X^{\text{corrupt}} = X + e$. Then the encoder produces latent representation Z from X^{corrupt}. The decoder can be formulated as:

$$\begin{aligned} \Pi &= \text{sigmoid}\left(W_\pi f_D(Z)\right) \\ M &= \text{diag}(s_i) \times \exp\left(W_\mu f_D(Z)\right) \\ \Theta &= \exp\left(W_\theta f_D(Z)\right) \end{aligned} \tag{21}$$

where $f_D(\cdot)$ is the decoder function; s_i is the size factor; $\{\Pi, M, \Theta\}$ are the estimations of ZINB distribution parameters $\{\pi, \mu, \theta\}$, respectively. The clustering process is based on soft assignment. The soft label q_{ij} of embedded point z_i is defined as:

$$q_{ij} = \frac{\left(1 + \|z_i - \mu_j\|^2\right)^{-1}}{\sum_k \left(1 + \|z_i - \mu_k\|^2\right)^{-1}} \tag{22}$$

where μ_j is the j-th cluster center.

dance.modules.single_modality.clustering.scdcc scDCC [132] shares the same model structure as scDeepCluster. In the training process, pairwise constraints are integrated into the loss function. There are two types of pairwise constraints, i.e., must-link (ML) and cannot-link (CL). Two instances with a must-link constraint should have similar soft labels as:

$$L_{\mathrm{ML}} = -\sum_{(a,b)\in ML} \log \sum_j q_{aj} \times q_{bj} \tag{23}$$

While two instances with cannot-link should have different soft labels as:

$$L_{\mathrm{CL}} = -\sum_{(a,b)\in CL} \log \left(\sum_j q_{aj} \times q_{bj} \right) \tag{24}$$

where q_{aj} and q_{bj} are soft labels defined in (22).

dance.modules.single_modality.clustering.graphsc graph-sc [26] utilizes gene-to-cell graph as the input of graph autoencoder. In the gene-to-cell graph, genes and cells are nodes, and there are weighted edges between cell nodes and the expressed gene nodes. Let the raw data matrix be X, then the weight of gene i to cell j is $w_{ij} = \frac{X[i,j]}{\sum_{k=0}^{m} X[k,j]}$. We use W, Z_0, and A to denote the graph weight matrix, the input features, and the adjacency matrix. $\bar{A} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ is the normalized adjacency matrix with D as the degree matrix of A. The cell embeddings can be obtained by:

$$Z = \mathrm{ReLU}(\mathrm{ReLU}(\bar{A} Z_0 W_1 W) W_2) \tag{25}$$

where W_1 and W_2 are learnable weights. The new adjacency matrix $\hat{A}$ is then reconstructed by $\hat{A} = \mathrm{sigmoid}(ZZ^\top)$.

dance.modules.single_modality.clustering.sctag scTAG [151] first generates a K-nearest neighbor cell-to-cell graph. It then adopts a ZINB-based graph autoencoder to process it, which takes topology adaptive graph convolutional network (TAGCN) [35] as the graph encoder. Consider the l-th hidden layer, let the input data be $x_c^{(l)} \in \mathbf{R}^N$, where $c = 1, 2, \ldots, C_l$, C_l is the number of features of each node, and N denotes the number of samples. The graph convolution process can be defined as follows:

$$x_f^{(l+1)} = \mathrm{ReLU}\left(\sum_{c=1}^{C_l} \sum_{k=0}^{K} g_{c,f,k}^{(l)} A^k x_c^{(l)} + b_f \mathbf{1}_N \right) \tag{26}$$

where b_f is a learnable bias; K is the number of convolution kernels; and $g_{c,f,k}^{(l)}$ denotes the polynomial coefficients. Denote Z as the latent embedded representation. The new adjacency matrix $\hat{A}$ is then reconstructed by $\hat{A} = \mathrm{sigmoid}(ZZ^\top)$. The estimations of ZINB distribution parameters $\{\pi, \mu, \theta\}$ are obtained by (21).

dance.modules.single_modality.clustering.scdsc scDSC [42] consists of a ZINB-based autoencoder and a graph autoencoder with the KNN cell-to-cell graph. In the ZINB-based autoencoder, let the latent representation be H, the output of last decoding layer be D, b_{enc} and b_{dec} be bias of encoder and decoder, respectively. The autoencoder is formulated as follows:

$$\begin{aligned} H &= f_{\text{enc}}(W_{\text{enc}}X + b_{\text{enc}}) \\ \bar{X} &= f_{\text{dec}}(W_{\text{dec}}H + b_{\text{dec}}) \end{aligned} \tag{27}$$

where $\bar{X}$ is the reconstructed expression matrix. The estimations of ZINB distribution parameters $\{\pi, \mu, \theta\}$ are similar to those in (21). In the graph encoder, denote the representation of l-th layer as $Z_{(l)}$, the adjacency matrix as A, and the degree matrix as D. The new representation is generated by:

$$Z_{(l+1)} = \phi\left(D^{-\frac{1}{2}}AD^{-\frac{1}{2}}\left(\sigma Z_{(l)} + (1-\sigma)H_{(l)}\right)W_{(l)}\right) \tag{28}$$

where $\phi(\cdot)$ is an activation function; σ is a hyperparameter; and $H_{(l)}$ is the representation of l-th layer of ZINB-based encoder.

D.2 Multimodality Module

Modality Prediction Modality prediction is to predict features of a target modality from features of an input modality. The evaluation is based on rooted mean squared error (RMSE) between ground-truth features and prediction. In this task, DANCE supports 4 models. All of them are deep learning models, one of which is based on graph neural networks.

dance.modules.multi_modality.predict_modality.scmogcn scMoGNN [143] first converts the input feature matrix into a cell-feature bipartite graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where each node v represents a cell or a feature. For instance, for the input of gene expression, each node can be a cell or a gene. Meanwhile, additional gene-gene connections are added based on an external pathway dataset [127]. Every pair of vertices in V are connected by a weighted edge, which either depends on the read count of a feature in a cell, or the correlation between features.

Specifically, we use $\mathbf{X} \in \mathbb{R}^{N\times k}$ to denote the feature matrix of input modality with N the number of cells and k the dimension of input features. The adjacency matrix $\mathbf{A}$ of the constructed cell-feature bipartite graph $\mathcal{G}$ is:

$$\mathbf{A} = \begin{pmatrix} \mathbf{O} & \mathbf{X} \\ \mathbf{X}^T & \mathbf{P} \end{pmatrix}, \tag{29}$$

where $\mathbf{O}$ is a zero matrix, and $\mathbf{P} \in \mathbb{R}^{k\times k}$ indicates the gene-gene links.

To learn node embeddings on $\mathcal{G}$, scMoGNN introduce a heterogeneous graph convolutional network. The proposed heterogeneous graph convolutional network can be stated as:

$$\mathbf{H}_{\text{c}}^{(l+1)} = \text{GCN}_{\text{fc}}(\mathbf{A}_{\text{fc}}, \mathbf{H}^{(l)}), \tag{30}$$

$$\mathbf{H}_{\mathrm{f}}^{(l+1)} = \mathrm{GCN}_{\mathrm{cf}}(\mathbf{A}_{\mathrm{cf}}, \mathbf{H}^{(l)}) + \mathrm{GCN}_{\mathrm{fc}}(\mathbf{A}_{\mathrm{ff}}, \mathbf{H}^{(l)}), \tag{31}$$

where $\mathbf{H}_{\mathrm{f}}^{(l)}$ is the embeddings of feature nodes in l-th layer, $\mathbf{H}_{\mathrm{c}}^{(l)}$ is the embeddings of cell nodes in l-th layer. Subscripts cf, $textfc$ and $textff$ denote the graph convolution over all the cell-to-feature edges, feature-to-cell edges and feature-to-feature edges respectively. After stacking convolutional layers, cell node embeddings from each layer are collected by a weighted sum, denoted as

$$\hat{\mathbf{H}} = \sum_{l=1}^{L} \mathbf{w}_l \mathbf{H}_c^l, \tag{32}$$

where $\mathbf{w}$ is a learnable weight vector, L is the total number of layers. $\hat{\mathbf{H}}$ is then passed to the downstream modules. In the case of modality prediction, a fully-connected predictive head is added. The final prediction $\mathbf{Z} \in \mathbb{R}^{N \times d}$ of scMoGNN can be thus written as:

$$\mathbf{Z} = \mathrm{ReLU}\left(\hat{\mathbf{H}}\mathbf{W} + \mathbf{b}\right) \tag{33}$$

where $\mathbf{W}$ and $\mathbf{b}$ are the parameters of the predictive head. Overall, a mean squared error $\mathrm{MSE}(\mathbf{Z}, \mathbf{Y})$ is optimized through training, where $\mathbf{Y} \in \mathbb{R}^{N \times d}$ is ground-truth features of the target modality.

dance.modules.multi_modality.predict_modality.babel BABEL [146] trains two neural-network-based encoders and two decoders on the paired data to translate data from one modality to the other and to reconstruct itself, thus eventually obtaining shared embedding. A special design in BABEL is to add a prior distribution to the decoder. Instead of directly outputting the feature values, the decoder estimates the parameters of feature distribution.

Formally, for each cell, the RNA decoder models the likelihood of expressions y as a negative binomial (NB) distribution, written as:

$$P(y; \hat{y}, \theta) = \frac{\Gamma(y+\theta)}{y!\Gamma(y)} \left(\frac{\theta}{\theta + \hat{y}}\right)^{\theta} \left(\frac{\hat{y}}{\theta + \hat{y}}\right)^{y} \tag{34}$$

where Γ denotes the gamma function, $\hat{y}$ and θ are the estimation of the mean and dispersion of the distribution. In practice, these estimations come from neural network encoders and decoders. The optimization problem is thus formalized by minimizing the negative log-likelihood:

$$\begin{aligned} \mathcal{L}_{\mathrm{NB}}(y; \hat{y}, \theta) = &-\theta(\log(\theta + \epsilon) - \log(\theta + \hat{y})) - y(\log(\hat{y} + \epsilon) - \log(\theta + \hat{y})) \\ &- \log \Gamma(y + \theta) + \log \Gamma(y + 1) + \log \Gamma(\theta + \epsilon) \end{aligned} \tag{35}$$

where ϵ is a tiny constant for numerical stability.

The ATAC decoder has a different modeling approach since the feature for each peak is binary. The loss function for the ATAC decoder is based on binary cross-entropy, shown below:

$$\mathcal{L}_{\mathrm{BCE}}(x; \widehat{x}) = -(x \log \widehat{x} + (1 - x) \log(1 - \widehat{x})) \tag{36}$$

where x represents ground-truth ATAC-seq features, and $\widehat{x}$ denotes the prediction from the ATAC decoder.

The overall loss function is hereby formulated as:

$$\mathcal{L} = \mathcal{L}_{\mathrm{NB}}\left(r, r_{\mathrm{RNA}}\right) + \beta \mathcal{L}_{\mathrm{BCE}}\left(a, a_{\mathrm{ATAC}}\right) + \beta \mathcal{L}_{\mathrm{BCE}}\left(a, a_{\mathrm{RNA}}\right) + \mathcal{L}_{\mathrm{NB}}\left(r, r_{\mathrm{ATAC}}\right) \quad (37)$$

where r and a are the ground-truth RNA-seq features and ATAC-seq features respectively, subscripts indicate from which modality the features are predicted (e.g., a_{RNA} represents the ATAC features predicted from RNA). The first two terms in the loss function are similar to reconstruction loss in autoencoders. The latter two terms can be considered as modality prediction loss.

For testing, we simply take a_{RNA} as the prediction of ATAC-seq from RNA-seq, and take r_{ATAC} as the prediction of RNA-seq from ATAC-seq.

dance.modules.multi_modality.predict_modality.cmae Cross-modal Autoencoders [149] uses autoencoders to map vastly different modalities (including images) to a shared latent space. Specifically, a discriminator and adversarial loss are added to force the distributions of different modalities to be matched in the latent space. To make use of prior knowledge, an additional loss term can further be added to align specific markers or anchoring cells.

Formally, an invariant latent distribution for two modalities i and j is learned as follows. We denote the input features of two modalities as $\mathbf{X}_i$ and $\mathbf{X}_j$. For modality i, we optimize the objective:

$$\min_{\mathrm{E}_i, \mathrm{D}_i} \mathbb{E}_{\mathbf{x} \sim P_{\mathbf{X}_i}} \mathcal{L}_1\left(\mathbf{x}, \mathrm{D}_i\left(\mathrm{E}_i(\mathbf{x})\right)\right) + \lambda \mathcal{L}_2\left(\mathrm{E}_i \# P_{\mathbf{X}_i} \mid P_{\hat{\mathbf{Z}}}\right) \quad (38)$$

while for modality j, we optimize:

$$\min_{\mathrm{E}_j, \mathrm{D}_j} \mathbb{E}_{\mathbf{x} \sim P_{\mathbf{X}_j}} \mathcal{L}_1\left(\mathbf{x}, \mathrm{D}_j\left(\mathrm{E}_j(\mathbf{x})\right)\right) + \lambda \mathcal{L}_2\left(\mathrm{E}_j \# P_{\mathbf{X}_j} \mid P_{\hat{\mathbf{Z}}}\right) \quad (39)$$

Here, $\mathrm{E}_i \# P_{\mathbf{X}_i}$ refers to the distribution of modality i in the latent space Z, $P_{\hat{\mathbf{Z}}_i}$ is the expected distribution of joint latent space Z. D and E refer to encoders and decoders parameterized by neural networks. $\mathcal{L}_1$ is the Euclidean distance metric, which is equivalent to a reconstruction loss. $\mathcal{L}_2$ represents a divergence between probability distributions, since $P_{\hat{\mathbf{Z}}_i}$ is unknown, it can be adapted to a discriminator and an adversarial loss.

Several additional losses can be added to the model to incorporate prior knowledge. For example, if training data include paired multimodal data, those cells with more than one modality can be considered as anchors. Suppose $\left(x_1, x_1'\right), \left(x_2, x_2'\right), \ldots, \left(x_m, x_m'\right)$ are corresponding points from two datasets, we can add the following anchor loss,

$$\sum_{i=1}^{m} ||\mathrm{E}\left(x_i\right) - \mathrm{E}'\left(x_i'\right)|| \quad (40)$$

where E and E$'$ are encoders for the two modalities, respectively.

The final prediction from modality i to modality j is $\mathrm{D}_j\left(\mathrm{E}_i(\mathbf{X})\right)$.

dance.modules.multi_modality.predict_modality.scmm scMM [92] leverages a mixture-of-experts (MoE) multimodal variational autoencoder [119] (VAE) to explore the latent dimensions that associate with multimodal regulatory programs. It models raw count features from each modality using various probability distributions in an end-to-end way. Specifically, an MoE multimodal VAE (MMVAE) is to learn a multimodal generative model:

$$p_{\Theta}\left(\boldsymbol{z}, \boldsymbol{x}_{1:M}\right)=p(\boldsymbol{z}) \prod_{m}^{M} p_{\theta_m}\left(\boldsymbol{x}_m \mid \boldsymbol{z}\right), \tag{41}$$

where $p_{\theta_m}\left(\boldsymbol{x}_m \mid \boldsymbol{z}\right)$ is the likelihood for m-th modality, and p_{θ_m} is parameterized by a neural network decoder. To optimize the model, a typical training objective for VAE is to maximize the ELBO:

$$\mathrm{ELBO}=\mathbb{E}_{z \sim q_{\Phi}}\left(z \mid \boldsymbol{x}_{1:M}\right)\left[\log \frac{p_{\Theta}\left(z, x_{1:M}\right)}{q_{\Phi}\left(z \mid x_{1:M}\right)}\right] \tag{42}$$

where $q_{\Phi}\left(\boldsymbol{z} \mid \boldsymbol{x}_{1:M}\right)$ is the joint variational posterior that can be parameterized by a neural network encoder. In addition, an MMVAE factorizes the joint posterior with an MoE:

$$q_{\Phi}\left(\boldsymbol{z} \mid \boldsymbol{x}_{1:M}\right)=\sum_{m}^{M} \alpha_m q_{\varphi_m}\left(\boldsymbol{z} \mid \boldsymbol{x}_m\right), \alpha_m=1/M, \tag{43}$$

where the posterior of the m-th modality is $q_{\varphi_m}\left(\boldsymbol{z} \mid \boldsymbol{x}_m\right)$. It is parameterized by the encoder. When using stratified sampling, ELBO can be re-written as:

$$\begin{aligned}
\mathrm{ELBO} &= \frac{1}{M} \sum_{m}^{M} \mathbb{E}_{\boldsymbol{z}_m \sim q_{\varphi_m}}\left(z \mid \boldsymbol{x}_m\right)\left[\log \frac{p_{\Theta}\left(z_m, \boldsymbol{x}_{1:M}\right)}{q_{\overline{\Phi}}\left(\boldsymbol{z}_m \mid \boldsymbol{x}_{1:M}\right)}\right] \\
&= \frac{1}{M} \sum_{m}^{M}\left\{\mathbb{E}_{\boldsymbol{z}_m \sim q_{\varphi_m}\left(\boldsymbol{z} \mid \boldsymbol{x}_{m_i}\right)}\left[\log p_{\Theta}\left(\boldsymbol{x}_{1:M} \mid \boldsymbol{z}_m\right)\right]-\mathrm{KL}\left[q_{\varphi_m}\left(\boldsymbol{z} \mid \boldsymbol{x}_m\right) \| p(\boldsymbol{z})\right]\right\}
\end{aligned} \tag{44}$$

The first expectation term is to measure the reconstruction performance. Here, latent variables of each modality would be used to reconstruct all modalities, including cross-modal translation. The second term is a regularization term forcing the variational posterior to be consistent the prior distribution $p(\boldsymbol{z})$, which is a Laplacian distribution in practice. In the modality prediction task, we take the cross-modal generation results as the prediction of the model.

Modality Matching The objective of the modality matching task is to identify the cell correspondence across modalities. To be concrete, we separate each modality of the jointly profiled dataset into a subset, and the order of cells in each subset is disturbed. In the training dataset, the cell correspondence labels between subsets are given. While in the testing data, the correspondence is not

given. The model needs to learn to identify cell correspondence from the labeled training data and evaluate it on the testing data.

To provide a more flexible protocol, the model output is adapted to a matching score matrix $\mathbf{S} \in \mathbb{R}^{n \times n}$, where n is the number of cells, $\mathbf{S}_{i,j}$ is the probability that cell i from one modality corresponds to cell j from the other modality. Therefore, $\mathbf{S}$ is a non-negative matrix where each row sums to 1. As metrics, we compute the average probability assigned to the correct matching. In this module, DANCE now supports 3 models. All of them are deep learning models, one of which is based on graph neural networks.

dance.modules.multi_modality.match_modality.scmogcn The overall structure of scMoGNN in the modality matching task is the same as in the modality prediction task. However, in the modality prediction task, the input is only one modality, while in the modality matching task, features of two modalities are given altogether. Therefore, scMoGMM constructs two graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ for two modalities respectively. The cell node embeddings are obtained in the same way as before, denoted as $\hat{\mathbf{H}}_1$ and $\hat{\mathbf{H}}_2$. Then a matching head is added, formulated as:

$$\mathbf{S} = \hat{\mathbf{H}}_1 \cdot \hat{\mathbf{H}}_2^T \tag{45}$$

where $\mathbf{S}$ is the desired output matrix of matching scores. For training, we calculate the cross entropy loss between $\mathbf{S}$ and a ground-truth matching matrix $\mathbf{M} \in \mathbb{R}^{n \times n}$, where $\mathbf{M}_{i,j} = 1$ only when cell i in modality 1 corresponds to cell j in modality 2. In addition, several auxiliary losses are added, including a reconstruction loss and a translation loss.

For testing, a bipartite matching via the Hungarian algorithm is implemented as post-processing. It generates an optimized sparse score matrix over the raw $\mathbf{S}$ matrix.

dance.modules.multi_modality.match_modality.cmae The overall structure of Cross-modal Autoencoders is the same as in the modality prediction task, where we implement encoders and decoders for all the modalities. Hereby in the modality matching task, we directly utilize the latent space instead of using a decoder to generate target modality.

To be specific, the embeddings from each modality can be denoted as:

$$\mathbf{H}_1 = \mathrm{E}_1\left(\mathbf{X}_1\right), \mathbf{H}_2 = \mathrm{E}_2\left(\mathbf{X}_2\right) \tag{46}$$

where we denote input matrix, encoder and embeddings of m-th modality as E_m, $\mathbf{H}_m$ and $\mathbf{X}_m$ respectively. Then the score matrix is obtained by:

$$\mathbf{S} = \mathbf{H}_1 \cdot \mathbf{H}_2^T \tag{47}$$

where $\mathbf{S}$ is the output score matrix.

dance.modules.multi_modality.match_modality.scmm The overall structure of scMM is the same as in the modality prediction task, where we implemented a neural network encoder E_m for each modality m to estimate the

variational posterior $q_\Phi(\boldsymbol{z} \mid \boldsymbol{x}_{1:M})$. In the modality matching task, we hereby take the latent vectors generated by encoders as the source for matching. The whole process is the same as Eq. 46 and 47.

Joint Embedding Joint embedding aims to encode features from two modalities into a low-dimensional joint latent space. To be consistent with the NeurIPS competition [80], we set the latent dimension size to 100. For the evaluation, currently, we only support NMI and ARI with the k-means clustering as metrics in our DANCE package. These metrics evaluate the consistency between latent clusters and the ground-truth cell type labels. More comprehensive metrics were introduced in the competition, and we are going to incorporate them into our package in the future. In this module, we now support 4 models. All of them are deep learning models, one of which is a graph neural network.

dance.modules.multi_modality.joint_embedding.scmogcn The overall structure of scMoGNN in the joint embedding task is still similar to what is shown in the modality prediction task. However, different from previous tasks, here scMoGNN first preprocesses data as suggested by Seurat [125]. It reduces the dimension of modality m to a predefined k_m dimension, using latent semantic indexing (LSI). Empirically, we set $k_m = 256$ for RNA-seq features, $k_m = 512$ for ATAC-seq features, and no dimension reduction for surface protein features. Next, the preprocessed features of two modalities are concatenated and jointly considered as feature nodes in the graph construction, as described in Eq. 29. Since no feature interactions are specified, here we replace matrix $\mathbf{P}$ in Eq. 29 with $\mathbf{O}$. With the graph $\mathcal{G}$ so constructed, scMoGNN is further trained by minimizing a reconstruction loss, a cell type auxiliary loss and a regularization loss.

Specifically, cell embeddings $\hat{\mathbf{H}}$ are obtained as Eq. 32. An MLP decoder f_θ is involved to reconstruct the input features from $\hat{\mathbf{H}}$. The first T dimensions in $\hat{\mathbf{H}}$ are also used to predict cell type, where T is equal to the number of predefined cell types, and a regularization loss is added to the rest of the dimensions. The overall loss function can be written as:

$$\begin{aligned}\mathcal{L} &= \mathcal{L}_{\text{recon}} + \mathcal{L}_{\text{cell type}} + \mathcal{L}_{\text{regular}} \\ &= \frac{1}{N}\sum_{i=1}^{N}(\mathbf{X} - f_\theta(\hat{\mathbf{H}}))^2 + \sum_{t=1}^{T}\mathbf{Y}_t \log(\hat{\mathbf{Y}}_t) + \beta * \|\hat{\mathbf{H}}_{\tilde{\mathcal{J}}}\|_2\end{aligned} \tag{48}$$

where f_θ is a two-layer MLP decoder, $\mathbf{X} \in \mathbb{R}^{n\times(k_1+k_2)}$ is the pre-processed feature matrix, k_1 and k_2 are specified feature dimensions of two modalities, $\mathbf{Y} \in \mathbb{R}^{N\times T}$ is predefined cell types for each cell in a sparse form, and $\hat{\mathbf{H}}_{\tilde{\mathcal{J}}}$ refers to the hidden dimensions other than first T dimensions. $\hat{\mathbf{Y}}$ is calculated by a softmax function over the first T dimensions of $\hat{\mathbf{H}}$, formulated as:

$$\hat{\mathbf{Y}}_{i,t} = \frac{e^{\hat{\mathbf{H}}_{i,t}}}{\sum_{k=1}^{T} e^{\hat{\mathbf{H}}_{i,k}}} \tag{49}$$

In the end, $\hat{\mathbf{H}}$ is the resulting joint cell embeddings from scMoGNN, which is expected to encode cellular information that is essential in the joint embedding task.

dance.modules.multi_modality.joint_embedding.jae JAE is an adapted model from scDEC [78]. It is proposed by the authors of scDEC in the NeurIPS competition [80] to better leverage cell annotations. Formally, JAE follows the typical autoencoder architecture with an encoder f_θ and a decoder g_θ. They are parameterized by MLPs, denoted as:

$$\mathbf{H} = f_\theta\left(\mathbf{X}\right), \mathbf{Z} = g_\theta\left(\mathbf{H}\right) \tag{50}$$

where $\mathbf{H} \in \mathbb{R}^{n\times d}$ is the joint embeddings, and $\mathbf{Z}$ is the recovered input features. The novelty of JAE is that it separates cell embeddings into four parts, written as:

$$\mathbf{H} = [\mathbf{H}'|\mathbf{C}|\mathbf{B}|\mathbf{S}] \tag{51}$$

where $|$ denotes concatenation, $\mathbf{C} \in \mathbb{R}^{n\times c}$ is additionally supervised by cell type labels, $\mathbf{B} \in \mathbb{R}^{n\times b}$ is additionally supervised by batch labels, $\mathbf{S} \in \mathbb{R}^{n\times s}$ is additionally supervised by cell cycle phase score, $\mathbf{H}' \in \mathbb{R}^{n\times z}$ is the remaining dimensions. Therefore $z + b + s + c = d$. The overall loss function of JAE is formulated as:

$$\begin{aligned}\mathcal{L} &= \mathcal{L}_{\text{recon}} + \mathcal{L}_{\text{cell_type}} + \mathcal{L}_{\text{batch}} + \mathcal{L}_{\text{cell_cycle}} \\ &= \text{MSE}(\mathbf{Z}, \mathbf{X}) + \text{CrossEntropy}(\mathbf{C}, \hat{\mathbf{C}}) \\ &\quad + \text{CrossEntropy}(\mathbf{B}, \hat{\mathbf{B}}) + \text{MSE}(\mathbf{SW} + \mathbf{b}, \hat{\mathbf{S}})\end{aligned} \tag{52}$$

where $\hat{\mathbf{C}}$, $\hat{\mathbf{B}}$, $\hat{\mathbf{S}}$ are cell type labels, batch labels and cell cycle scores, respectively. $\mathbf{W} \in \mathbb{R}^{s\times 2}$ and $\mathbf{b} \in \mathbb{R}^2$ are an extra linear transformation to project $\mathbf{S}$ to cell cycle score vector space. Eventually, $\mathbf{H}$ is the output joint embedding from JAE.

dance.modules.multi_modality.joint_embedding.scmvae scMVAE [157] learn the distribution of multi-omics via three learning strategies simultaneously: product of experts (PoE), neural networks, and concatenation of multi-omics features. In addition, scMVAE models raw count features from each modality through a ZINB distribution. Specifically, let z be the joint embeddings obtained from multimodal encoders. $p(z|c)$ is a Gaussian mixture distribution. Its mean vector μ_c and a covariance matrix σ_c are conditioned on c, where c is a discrete categorical variable indicating cell types. This Gaussian mixture prior of z is introduced in MVAE to enhance the interpretability of the latent representations and the performance of generation. Let x and y be features of two modalities. Then the distribution $p(x, y, z, c, l_x, l_y)$ is formulated as:

$$p\left(x, y, z, c, l_x, l_y\right) = p\left(x \mid z, l_x\right) p\left(y \mid z, l_y\right) p(z \mid c) p(c) p\left(l_x\right) p\left(l_y\right) \tag{53}$$

where l_x and l_y are the library size factors of two modalities.

The prior distributions for all the random variables are listed below:

$$
\begin{aligned}
& z \sim N\left(\mu_c, \sigma_c^2 I\right) \\
& l_x \sim \log \operatorname{norm}\left(\mu_{lx}, \sigma_{lx}^2\right), l_y \sim \log \operatorname{norm}\left(\mu_{ly}, {\sigma_{ly}}^2\right) \\
& \mu_x \sim \operatorname{Gamma}\left(f_{\mu x}(f(z)), f_{\theta x}(f(z))\right), \mu_y \sim \operatorname{Gamma}\left(f_{\mu y}(f(z)), f_{\theta y}(f(z))\right) \\
& x' \sim \text{ Possion } \left(l_x \mu_x\right), y' \sim \text{ Possion } \left(l_y \mu_y\right) \\
& \pi_x \sim \operatorname{Bernoulli}\left(f_{\pi x}(f(z))\right), \pi_y \sim \operatorname{Bernoulli}\left(f_{\pi y}(f(z))\right) \\
& x_r = \begin{cases} x' \text{ if } \pi_x = 0 \\ 0 \text{ otherwise} \end{cases}, y_r = \begin{cases} y' \text{ if } \pi_y = 0 \\ 0 \text{ otherwise} \end{cases}
\end{aligned}
\tag{54}
$$

where $f_{\theta x}(f(z))$ and $f_{\theta y}(f(z))$ are the inverse desperations of two modalities from the variational Bayesian inference. $f_{\mu x}$ and $f_{\mu y}$ are two neural network decoders that estimate the mean proportions of features for two modalities in each cell by using a softmax function, which simulates the library-size normalized features. $f_{\pi x}$ and $f_{\pi y}$ are neural network decoders that estimate the probability of features being dropped out due to technical issues. They use a sigmoid function to model this probability.

To train scMVAE, we maximize the log-likelihood of the multi-omics observations. Following the convention of variational autoencoders, this objective is converted to optimizing an evidence lower bound (ELBO):

$$
\begin{gathered}
\log p\left(x, y \mid z, c, l_x, l_y\right) \geq E_{q_\varphi}\left(z, c, lx, ly \mid x, y\right)\big) \\
\left[\lambda_1 \log\left(p_{\theta_1}\left(x \mid z, l_x\right)\right) + \lambda_2 \log\left(p_{\theta_2}\left(y \mid z, l_y\right)\right)\right] \\
-\alpha_1 D_{KL}\left(q\left(l_x \mid x\right) \| p\left(l_x\right)\right) - \alpha_2 D_{KL}\left(q\left(l_y \mid y\right) \| p\left(l_y\right)\right) \\
-\beta D_{KL}(q(z, c \mid x, y) \| p(z, c))
\end{gathered}
\tag{55}
$$

Both modalities in the ELBO have two reconstruction terms, and three regularization terms are implemented by KL divergence. q_φ refers to a multimodal encoder. While p_{θ_1} and p_{θ_2} refer to decoders, for two modalities respectively. The latent vector z estimated from E is the eventual joint embedding.

dance.modules.multi_modality.joint_embedding.dcca In DCCA [157], data of each modality are modeled by a variational autoencoder (VAE). Specifically, for modality m, an encoder E_m transforms the input features into latent space z_m. A decoder D_m then transforms z_m into the parameters of the NB or Bernoulli. For example, RNA-seq and ADT data follow NB distribution, denoted as:

$$
\begin{gathered}
p\left(x \mid z_x\right) = \operatorname{NB}\left(x; u_x, \theta_x\right) = \operatorname{NB}\left(x, l_x; D_{u_x}\left(z_x\right), D_{\theta_x}\left(z_x\right)\right) \\
\operatorname{NB}\left(x; u_x, \theta_x\right) = \frac{\Gamma\left(x+\theta_x\right)}{\Gamma\left(\theta_x\right) \Gamma(x+1)}\left(\frac{u_x}{u_x+\theta_x}\right)^x\left(\frac{\theta_x}{\theta_x+u_x}\right)^{\theta_x}
\end{gathered}
\tag{56}
$$

where D_{u_x} and D_{θ_x} are decoders, each dimension of u_x and θ_x indicates the mean and variance of NB distribution for each feature, and one-dimensional constant variable l_x indicates the library size of each cell. While ATAC-seq data follow

Bernoulli distribution, denoted as:

$$\begin{gathered} p\left(y \mid z_y\right)=\text{Bernoulli}\left(y ; u_y\right)=\text{Bernoulli}\left(y ; D_{u_y}\left(z_y\right)\right) \\ \text{Bernoulli}\left(y ; u_y\right)=y \log \left(u_y\right)+(1-y) \log \left(1-u_y\right) \end{gathered} \tag{57}$$

where D_{uy} refers to the ATAC-seq decoder.

Each VAE is first trained separately with each modality. Then, two VAEs are trained together to maximize the similarity between two latent spaces. For example, given the embeddings from a VAE_{RNA} and a VAE_{ATAC}, they optimize an objective function that combines reconstruction loss with the cell embeddings similarity loss. Hence, the total ELBO for VAE_{RNA} and VAE_{ATAC} can be written as:

$$\begin{aligned} \mathcal{L}_{\text{RNA}} &= \log p\left(x \mid z_x\right)-\beta_1 \sum_{i=0}^{K-1}\left\|{z_y}^i-{z_x}^i\right\|_2 \\ &\geq E_{z_x \sim q\left(z_x \mid x ; E_x\right)}\left(\log p\left(x \mid z_x ; D_{ux}, D_{\theta x}\right)\right)-\lambda_1 D_{KL}\left(q\left(z_x \mid x ; E_x\right)\|p(z)\right) \\ &\quad -\beta_1 \sum_{i=0}^{K-1}\left\|{z_y}^i-{z_x}^i\right\|_2 \end{aligned} \tag{58}$$

$$\begin{aligned} \mathcal{L}_{\text{ATAC}} &= \log p\left(y \mid z_y\right)-\beta_2 \sum_{i=0}^{K-1}\left\|{z_x}^i-{z_y}^i\right\|_2 \\ &\geq E_{z_y \sim q\left(z_y \mid y ; E_y\right)}\left(\log p\left(y \mid z_y ; D_{uy}\right)\right)-\lambda_2 D_{KL}\left(q\left(z_y \mid y ; E_y\right)\|p(z)\right) \\ &\quad -\beta_2 \sum_{i=0}^{K-1}\left\|{z_x}^i-{z_y}^i\right\|_2 \end{aligned} \tag{59}$$

According to the DCCA paper, after jointly training two VAEs, the embedding from VAE_{RNA} is selected as the final embedding for downstream analysis.

D.3 Spatial Transcriptomics Module

Spatial Domain In spatial transcriptomics, the spatial data is referring to spots with x,y coordinates, and each spot captures several cells. The objective of the spatial domain is to partition the spatial data into meaningful clusters. Each cluster uncovered by this analysis is regarded as a spatial domain. Spots in the same spatial area are comparable and consistent in gene expression and histology, but spots in different spatial regions are distinct [13]. For evaluation, Adjusted Rand Index (ARI) [150] is utilized to compare the efficacy of various clustering techniques. It computes the similarity between the algorithm-predicted clustering labels and the actual labels. In the spatial domain task, DANCE supports 4 models including 2 GNN-based models and 2 traditional models.

dance.modules.spatial.spatial_domain.spagcn SpaGCN [57] first constructs

a weighted undirected graph, $G(V, E)$ from the gene expression and histological image data. In G, each vertex $v \in V$ is a spot, and every pair of vertices in V are connected by a weighted edge, which assesses the correlation between the two spots. The weight for the pair (u,v) is calculated as:

$$w(u, v) = \exp\left(-\frac{d(u, v)^2}{2l^2}\right) \tag{60}$$

where hyperparameter l represents characteristic length scale, and $d(u, v)$ calculates Euclidean distance between spots u and v. This distance is computed by the spatial distance and the corresponding histology information between two spots. The initial node representation in the graph is gene expression after dimension reduction. A process known as graph convolution is utilized by SpaGCN to aggregate gene expression data in accordance with edge weights. Then the output of the graph convolution layer would be new node representation capturing information on gene expression, histology and physical location. Based on the new spot representation, an unsupervised clustering algorithm is further employed to iteratively cluster the spots into spatial domains. The probability of assigning spot i to cluster j is defined as:

$$q_{ij} = \frac{\left(1 + h_i - \mu_j^2\right)^{-1}}{\sum_{j'=1}^{K}\left(1 + h_i - \mu_{j'}^2\right)^{-1}} \tag{61}$$

where h_i is the embedded point for spot i, and u_j indicates centroid j. Then the clusters are refined iteratively by a target distribution P from q_{ij}:

$$p_{ij} = \frac{q_{ij}^2 / \sum_{i=1}^{N} q_{ij}}{\sum_{j'=1}^{K}\left(q_{ij'}^2 / \sum_{i=1}^{N} q_{ij'}\right)} \tag{62}$$

which gives more weight to locations that have been recognized with high confidence and normalizes each centroid's contribution to the loss function to avoid having large clusters distort the hidden feature space. The objective function is based on a Kullback–Leibler (KL) divergence as:

$$L = KL(P\|Q) = \sum_{i=1}^{N}\sum_{j=1}^{K} p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{63}$$

where N is the number of spot samples, and K is the number of clusterings.

dance.modules.spatial.spatial_domain.stagate STAGATE [33] is a graph attention based autoencoder [116] with encoder, decoder and graph attention layers. For graph construction, it builds up an undirected graph with a radius r that has been predefined based on physical distance. We use$\mathbf{A}$ to denote the adjacency matrix of the graph where $\mathbf{A_{ij}} = 1$ if the Euclidean distance between

spots i and j is less than r. In addition, STAGATE also builds up a cell type-aware graph via updating the previously constructed graph based on the pre-clustering of gene expressions.

The encoder in STAGATE takes the gene expressions that have been normalized as its inputs. It then generates embedding for each spot by aggregating information from its surrounding nodes collectively. The latent representation for spot i is calculated as:

$$\mathbf{h}_i^{(k)} = \sigma\left(\sum_{j \in S_i} \mathbf{att}_{ij}^{(k)} \left(\mathbf{W}_k \mathbf{h}_j^{(k-1)}\right)\right) \tag{64}$$

where σ is the activation function, $\mathbf{W}_k$ is the trainable weight matrix, S_i is the neighbors of spot i, k indicates k-th encoder layer and $\mathbf{att}_{ij}^{(k)}$ is the attention score (i.e., the edge weight) between spots i and j obtained from the k-th graph attention layer's output.

Contrarily, using the encoder's output as input, the decoder transforms the latent embedding into a reconstructed normalized expression profile.

$$\widehat{\mathbf{h}}_i^{(k-1)} = \sigma\left(\sum_{j \in S_i} \widehat{\mathbf{att}}_{ij}^{(k-1)} \left(\widehat{\mathbf{W}}_k \widehat{\mathbf{h}}_j^{(k)}\right)\right) \tag{65}$$

In graph attention layer, a widely-used self-attention technique for graph neural networks is adopted to learn the similarity between surrounding spots in an adaptive manner. The edge weight between spot i and its surrounding spot j in the k-th encoder layer is calculated as:

$$e_{ij}^{(k)} = \text{Sigmoid}\left(\mathbf{v}_s^{(k)^T}\left(\mathbf{W}_k \mathbf{h}_i^{(k-1)}\right) + \mathbf{v}_r^{(k)^T}\left(\mathbf{W}_k \mathbf{h}_j^{(k-1)}\right)\right) \tag{66}$$

where $\mathbf{v}_s^{(k)}$ and $\mathbf{v}_r^{(k)}$ are the trainable weights.

The attention score is further normalized by a softmax function in the following way:

$$att_{ij}^{(k)} = \frac{\exp\left(e_{ij}^{(k)}\right)}{\sum_{i \in \mathscr{S}_i} \exp\left(e_{ij}^{(k)}\right)} \tag{67}$$

Eventually, STAGATE aims to minimize the reconstruction loss of normalized expressions in the following way:

$$\sum_{i=1}^{N} \left\| \mathbf{x}_i - \widehat{\mathbf{h}}_i \right\|_2 \tag{68}$$

where $\mathbf{x}_i$ is the original normalized gene expressions, and $\widehat{\mathbf{h}}_i$ is the reconstructed normalized gene expression for spot i.

dance.modules.spatial.spatial_domain.louvain Louvain [11] is to extract

the community structure of large networks, which draws inspiration from the optimization of modularity. A modularity measure shows how densely edges inside communities are clustered in comparison to edges outside communities. Theoretically, optimizing this value leads to the optimal grouping of nodes in a particular network. Due to the impracticality of traversing all possible iterations of the nodes into groups, heuristic approaches are utilized. The modularity is defined as:

$$Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta \left(c_i, c_j \right) \tag{69}$$

where A_{ij} is the edge weight between nodes i and j, m represents the total weight of all edges in the graph, the weights of the edges that connect nodes i and j are added up to get k_i and k_j, c_i and c_j denote node communities, and δ represents Kronecker delta function [9] ($\delta(x, y) = 1$ if $x = y$; else, 0).

To efficiently maximize the modularity, there are two looping phases. First, a community is associated with each node in the network. Because of this primary partitioning, there is consequently the same number of communities as nodes. Then, the change in modularity is determined for each node i by removing it from its own community and inserting it in the community of each of i's neighbors j. Integration of a previously isolated node i into a community C results in an increase in modularity ΔQ equal to:

$$\Delta Q = \left[\frac{\Sigma_{in} + 2k_{i,in}}{2m} - \left(\frac{\Sigma_{tot} + k_i}{2m} \right)^2 \right] - \left[\frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m} \right)^2 - \left(\frac{k_i}{2m} \right)^2 \right] \tag{70}$$

where $\sum_{tot}$ is the total link weights occurring to nodes in C, $\sum_{in}$ represents the total link weights within C, k_i denotes the sum of the link weights associated with node i, $k_{i,in}$ represents the total link weights from node i to all other nodes in C, and m is the total link weights for the whole network. Once this value is computed for all communities to which i belongs, i is assigned to the community where the modularity increase was largest. If there is no possibility of expansion, i stays in the same community. This technique is repeated and applied successively to all nodes until no further growth in modularity is possible. The first step concludes when this modularity maximum is reached.

In the second phase, all nodes inside the same community are grouped together and a new network consisting of communities from the first phase is constructed. The connections between multiple nodes within the same community and a node in a separate community are now described by weighted edges between communities. The second stage is complete when the new network is set up, at which point the first stage can be applied to it again.

dance.modules.spatial.spatial_domain.stlearn stLearn [109] performs unsupervised clustering on SME-normalized data to group similar areas into clusters and discover sub-clustering alternatives based on the geographic separation of clusters inside the tissue. The name of this stLearn function is SMEcluster. Using normalized expression values, stLearn separates cell types in a tissue through

a two-step spatial clustering approach. stLearn implements a conventional Louvain clustering technique for scRNAseq data as the initial step. Linear Principal Component Analysis (PCA) is used to reduce the dimensionality of the SME normalized matrix, and then non-linear UMAP embedding is used to generate the k-nearest neighbor (kNN) graph. kNN's graph adjacency matrix is then clustered using Louvain clustering or k-means clustering. In the second stage, spatial information is utilized to identify sub-clusters from large clusters that span two or more physically distinct places. Using each location's spatial coordinates, a two-dimensional k-d tree neighbour search is conducted.

Cell Type Deconvolution Cell-type deconvolution is the task of estimating cell-type composition in cell-pools from their aggregate transcriptomic information. This is a type of inverse problem, as we are trying to determine the signal of individual cell-types from aggregated readings across multiple cell-types. Moreover, due to the nature of the spatial (or bulk) transcriptomics profiling technologies, the true cell-type compositions are most often not given. Solving this task then requires some transfer learning approaches, in most cases using scRNA-seq data as a reference (transfer source).

The problem is formulated as follows. We're given mixed-cell expression data $X \in \mathbb{R}^{d \times n}$ where each cell-pool $i \in [1, n]$ is composed of a mixture of cell-types $[1, K]$. Then for each cell-pool $i \in [1, n]$, we wish to construct an estimator $\hat{y}_i \in \Delta^{K-1}$ of the true cell-type composition $y_i \in \Delta^{K-1}$. Here, Δ^{K-1} is the regular K-simplex

$$\Delta^{K-1} = \{x \in \mathbb{R}^K : \sum_{k=1}^{K} x_k = 1, x_k \geq 0 \text{ for } k = 1, 2, ..., K\} \tag{71}$$

We are also given some reference scRNA-seq expression data $X_s \in \mathbb{R}^{d \times N}$ with one-hot labeled cell-types $C \in \mathbb{R}^{N \times K}$. Then construct the estimator of cell-type compositions for the n cell-pools by some function

$$\widehat{Y} = F(X, X_s, C) \in \mathbb{R}^{n \times K} \tag{72}$$

If the spatial information (2D or 3D coordinates in the given tissue)

$$S = \begin{bmatrix} s_1 \\ \vdots \\ s_n \end{bmatrix} \in \mathbb{R}^{n \times m},\ m \in \{2, 3\}$$

of the cell-pools are available, then we may incorporate this into the estimator of cell-type compositions for the n cell-pools by some function

$$\widehat{Y} = F(X, S, X_s, C) \in \mathbb{R}^{n \times K} \tag{73}$$

For the task of cell-type deconvolution, DANCE supports 4 models, one GNN based model and 3 non-GNN based models with classical regression models as their backbone.

dance.modules.spatial.cell_type_deconvo.dstg DSTG [120] is a GNN based method that constructs a graph from similarities between real mixed-cell expression data $X \in \mathbb{R}^{d \times n}$ and pseudo mixed-cell expression data $\widetilde{X} \in \mathbb{R}^{d \times n_p}$ from reference scRNA-seq expression data $X_s \in \mathbb{R}^{d \times N}$. First, the pseudo mixed-cell expression data is generated taking n_p random samples (with replacement) of 2 to 8 cells from the scRNA-seq reference, and aggregating their UMI counts, downsampling to adjust for realistic bulk UMI counts. The pseudo and real mixed-cell data are then aligned in a lower dimensional ($S < d$) gene-space using Canonical Correlation Analysis (CCA). The projections to the $s = 1, 2, ..., S$ dimensions are given by the canonical variables

$$\begin{aligned} U_s &= \widetilde{X}\mu_s^* \\ V_s &= X\nu_s^* \end{aligned} \tag{74}$$

where

$$\mu_s^*, \nu_s^* = \underset{\mu_s, \nu_s \in \mathbb{R}^d}{argmax}\{\nu_s^T \widetilde{X}^T X \nu_s\} \text{ s.t. } U_s^T U_{s'} = V_s^T V_{s'} = \delta_{ss'} \tag{75}$$

are the canonical correlation vector pairs. These embeddings are then used to construct a graph by considering Mutual Nearest Neighbors (MNN) as adjacent in the graph. That is, given a pair of sample cell-pools i, j, we let

$$A_{i,j} = \begin{cases} 1 & \text{if } i \text{ and } j \text{ are mutual nearest neighbors} \\ 0 & \text{otherwise} \end{cases} \tag{76}$$

Here, adjacencies can be between simulated-to-real and real-to-real samples. With $X_{in} = [\widetilde{X} X] \in \mathbb{R}^{d \times N}$ ($N = n_p + n$) and the normalized adjacency matrix $\widetilde{A}$ as input, the $L \geq 1$ (default 1) graph convolution (GCN) layers of the DSTG model are given by

$$\begin{aligned} H^{(0)} &= X_{in} \\ H^{(l)} &= \text{ReLU}(\widetilde{A} H^{(l-1)} W^{(l)}) \text{ for } l \in [1, L] \end{aligned} \tag{77}$$

where $W^{(l)}$ is the weight matrix for the l_{th} layer. The output of the DSTG model is the predicted composition of K cell-types, given by

$$\begin{bmatrix} \hat{Y}_p \\ \hat{Y} \end{bmatrix} = \text{softmax}(\widetilde{A} H^{(L)} W) \in \mathbb{R}^{N \times K} \tag{78}$$

where $\hat{Y}_p$ and $\hat{Y}$ are the predictions for the pseudo and real cell-pools, respectively. The loss function is then defined as the cross-entropy between the predicted and true cell-type compositions of the pseudo cell-pools

$$\mathcal{L} = -\sum_{i=1}^{n_p} \sum_{k=1}^{K} y_{i,k}^{(p)} ln(\hat{y}_{i,k}^{(p)}) \tag{79}$$

where $\hat{y}_{i,k}^{(p)}$ and $y_{i,k}^{(p)}$ are the predicted and true composition of cell-type k in the i_{th} pseudo cell-pool.

dance.modules.spatial.cell_type_deconvo.spotlight SPOTlight [38] builds on the classic non-negative least squares approach to cell-type deconvolution by incorporating topic-modeling for both the reference scRNA-seq expression data $X_s \in \mathbb{R}^{d \times N}$ and the mixed-cell expression data $X \in \mathbb{R}^{d \times n}$. First, non-negative matrix factorization (NMF) is applied to the reference X_s to get

$$W, H = \underset{W', H' \geq 0}{argmin} \left\| X_s - W'H' \right\|_F \tag{80}$$

where the rows of $H \in \mathbb{R}^{K \times N}$ are the cell-topic embeddings, and the columns of $W \in \mathbb{R}^{d \times K}$ the corresponding weightings. Cell-topic profiles $\widetilde{H} \in \mathbb{R}^{K \times K}$ are then constructed from H by taking the median over each cell-type. Next, spot-topic profiles $P \in \mathbb{R}^{K \times n}$ are constructed through NNLS of X onto W

$$P = \underset{P' \geq 0}{argmin} \|X - WP'\|_F \tag{81}$$

Finally, the estimator of cell-type compositions for the n cell-pools is then given by

$$\widehat{Y} = \underset{B \geq 0}{argmin} \|P - \widetilde{H}B\|_F \tag{82}$$

dance.modules.spatial.cell_type_deconvo.spatialdecon SpatialDecon [30] is non-negative linear regression based method that assumes a log-normal multiplicative error model between the mixed-cell data $X \in \mathbb{R}^{d \times n}$ and a cell-profile (signature) matrix $\widetilde{X} \in \mathbb{R}^{d \times K}$. The cell-profile matrix $\widetilde{X}$ is a measure of center (median by default) expression for each of the K cell-types, constructed from the reference scRNA-seq data $X_s \in \mathbb{R}^{d \times N}$. The log-normal multiplicative error model is given by

$$log(X_{i\cdot}) = log(\widetilde{X}_{i\cdot}^T B) + \epsilon_i, \text{ where } \epsilon_i \sim \mathcal{N}(0, \sigma^2 I_n) \text{ and } B \in \mathbb{R}^{K \times n} \tag{83}$$

The estimator of cell-type compositions for the n cell-pools is then given by

$$\hat{Y} = \underset{B \geq 0}{argmin} \left\| log(X) - log(\widetilde{X}^T B) \right\|_2 \tag{84}$$

dance.modules.spatial.cell_type_deconvo.card CARD [85] applies a conditional autoregressive (CAR) assumption on the coefficients of the classical non-negative linear model between the mixed-cell expression X and a cell-profile matrix $\widetilde{X}_s$, constructed from reference scRNA-seq X_s. The linear model is given by

$$X = \widetilde{X}_s B + \epsilon, \epsilon \sim \mathcal{N}(0, \sigma_e^2 I_n) \tag{85}$$

The CAR assumption then incorporates 2D spatial information $S = \begin{bmatrix} s_1 \\ \vdots \\ s_n \end{bmatrix} \in \mathbb{R}^{n \times 2}$ through an intrinsic prior on the cell-type compositions (the

model coefficients) by modeling compositions in each location as a weighted combination of compositions in all other locations. This modeling assumption is given by

$$B_{ki} = b_k + \phi \sum_{j=1, j \neq i}^{n} W_{ij}(B_{kj} - b_k) + \epsilon_{ki}, \epsilon_{ki} \sim \mathcal{N}(0, \sigma_{ki}^2) \tag{86}$$

where the weights W_{ij} are given by the Gaussian kernel

$$W_{ij} = K_G(s_i, s_j; \sigma^2) = \exp(-\frac{\|s_i - s_j\|_2^2}{2\sigma^2}) \tag{87}$$

with default scaling parameter $\sigma^2 = 0.1$. CARD then estimates the cell-type composition of the n cell-pools through constrained maximum likelihood estimation.

E Structure Tree

The structure tree below shows our codebase structure in DANCE. Basically it consists of two parts: dance source code and examples. In example folder, each file represent one example to show how to leverage one specific model on dataset.

```
|___     LICENSE
|___     README.md
|___     dance
|   |___     __init__.py
|   |___     data
|   |   |___    __init__.py
|   |___     datasets
|   |   |___     __init__.py
|   |   |___     multimodality.py
|   |   |___     singlemodality.py
|   |   |___    spatial.py
|   |___     modules
|   |   |___     __init__.py
|   |   |___     multi_modality
|   |   |   |___     __init__.py
|   |   |   |___     joint_embedding
|   |   |   |   |___     __init__.py
|   |   |   |   |___     dcca.py
|   |   |   |   |___     jae.py
|   |   |   |   |___     scmogcn.py
|   |   |   |   |___     scmogcnv2.py
|   |   |   |   |___    scmvae.py
|   |   |   |___     match_modality
|   |   |   |   |___     __init__.py
|   |   |   |   |___     cmae.py
|   |   |   |   |___     scmm.py
|   |   |   |   |___    scmogcn.py
|   |   |   |___    predict_modality
|   |   |       |___     __init__.py
|   |   |       |___     babel.py
|   |   |       |___     cmae.py
|   |   |       |___     scmm.py
|   |   |       |___    scmogcn.py
|   |   |___     single_modality
|   |   |   |___     __init__.py
|   |   |   |___     cell_type_annotation
|   |   |   |   |___     __init__.py
|   |   |   |   |___     actinn.py
|   |   |   |   |___     celltypist.py
|   |   |   |   |___     scdeepsort.py
```

```
|   |   |   |   |___     singlecellnet.py
|   |   |   |   |___    svm.py
|   |   |   |___     clustering
|   |   |   |   |___     __init__.py
|   |   |   |   |___     graphsc.py
|   |   |   |   |___     scdcc.py
|   |   |   |   |___     scdeepcluster.py
|   |   |   |   |___     scdsc.py
|   |   |   |   |___    sctag.py
|   |   |   |___    imputation
|   |   |       |___     __init__.py
|   |   |       |___     deepimpute.py
|   |   |       |___     graphsci.py
|   |   |       |___    scgnn.py
|   |   |___    spatial
|   |       |___     __init__.py
|   |       |___     cell_type_deconvo
|   |       |   |___     __init__.py
|   |       |   |___     card.py
|   |       |   |___     dstg.py
|   |       |   |___     spatialdecon.py
|   |       |   |___    spotlight.py
|   |       |___    spatial_domain
|   |           |___     __init__.py
|   |           |___     louvain.py
|   |           |___     spagcn.py
|   |           |___     stagate.py
|   |           |___    stlearn.py
|   |___     plotting
|   |   |___    __init__.py
|   |___     transforms
|   |   |___     __init__.py
|   |   |___     graph_construct.py
|   |   |___    preprocess.py
|   |___    utils
|       |___     __init__.py
|       |___     loss.py
|       |___    metrics.py
|___     docs
|   |___     Makefile
|   |___     make.bat
|   |___     requirements.txt
|   |___    source
|       |___     conf.py
|       |___     index.rst
```

```
|        |___    modules
|            |___     multi_modality.rst
|            |___     single_modality.rst
|            |___    spatial.rst
|___     examples
|    |___     multi_modality
|    |    |___     joint_embedding
|    |    |    |___     dcca.py
|    |    |    |___     jae.py
|    |    |    |___     scmogcn.py
|    |    |    |___     scmogcnv2.py
|    |    |    |___    scmvae.py
|    |    |___     match_modality
|    |    |    |___     cmae.py
|    |    |    |___     scmm.py
|    |    |    |___    scmogcn.py
|    |    |___    predict_modality
|    |        |___     babel.py
|    |        |___     cmae.py
|    |        |___     scmm.py
|    |        |___    scmogcn.py
|    |___     single_modality
|    |    |___     cell_type_annotation
|    |    |    |___     actinn.py
|    |    |    |___     celltypist.py
|    |    |    |___     scdeepsort.py
|    |    |    |___     singlecellnet.py
|    |    |    |___    svm.py
|    |    |___     clustering
|    |    |    |___     graphsc.py
|    |    |    |___     scdcc.py
|    |    |    |___     scdeepcluster.py
|    |    |    |___     scdsc.py
|    |    |    |___    sctag.py
|    |    |___    imputation
|    |        |___     deepimpute.py
|    |        |___     graphsci.py
|    |        |___    scgnn.py
|    |___    spatial
|        |___     cell_type_deconvo
|        |    |___     card.py
|        |    |___     dstg.py
|        |    |___     spatialdecon.py
|        |    |___    spotlight.py
|        |___    spatial_domain
```

```
|            |___     louvain.py
|            |___     spagcn.py
|            |___     stagate.py
|            |___    stlearn.py
|___      imgs
|   |___     dance_logo.jpg
|   |___    framework.png
|___      install.sh
|___      pyproject.toml
|___      requirements.txt
|___      setup.cfg
|___      setup.py
|___      tests
|   |___    test_bench.py
|___    tox.ini
```