Ding *et al. Genome Biology* (2024) 25:72
https://doi.org/10.1186/s13059-024-03211-z

SOFTWARE

Open Access

DANCE: a deep learning library and benchmark platform for single-cell analysis

Jiayuan Ding[1*†], Renming Liu[2†], Hongzhi Wen[1†], Wenzhuo Tang[3†], Zhaoheng Li[4], Julian Venegas[2], Runze Su[2,3], Dylan Molho[2], Wei Jin[1], Yixin Wang[6], Qiaolin Lu[8], Lingxiao Li[7], Wangyang Zuo[5], Yi Chang[8], Yuying Xie[2,3*] and Jiliang Tang[1*]

†Jiayuan Ding, Renming Liu, Hongzhi Wen and Wenzhuo Tang contributed equally to this work.

*Correspondence: dingjia5@msu.edu; xyy@msu.edu; tangjili@msu.edu

[1] Department of Computer Science and Engineering, Michigan State University, East Lansing, USA
[2] Department of Computational Mathematics, Science and Engineering, Michigan State University, East Lansing, USA
[3] Department of Statistics and Probability, Michigan State University, East Lansing, USA
[4] Department of Biostatistics, University of Washington, Seattle, USA
[5] Department of Computer Science, Zhejiang University of Technology, Zhejiang, China
[6] Department of Bioengineering, Stanford University, Palo Alto, USA
[7] Department of Computer Science, Boston University, Boston, USA
[8] School of Artificial Intelligence, Jilin University, Jilin, China

Abstract

DANCE is 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, such as using only one command line. In addition, DANCE provides an ecosystem of deep learning architectures and tools for researchers to facilitate their own model development. DANCE is an open-source Python package that welcomes all kinds of contributions.

Keywords: Deep learning, Benchmarking, Single-cell multimodal analysis, Single-cell spatial analysis, Gene imputation, Cell type annotation, Clustering, Multimodality integration, Spatial domain identification, Cell type deconvolution

Background

Single-cell profiling technology has undergone rapid development in recent years, spanning from single modality profiling (RNA, protein, and open chromatin) [1–9], multimodal profiling [10–14], to spatial transcriptomics [15–22]. 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 (GNNs) [11, 23–28] have shown promising performance in single-cell analysis, the customized interfaces of such tools are largely missing in the existing packages. Those motivate the development of our DANCE system which 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 platform 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. 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 [29], Deep Graph Library (DGL) [30], and PyTorch Geometric (PyG) [31] 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 hyperparameters are saved into only one command line for user reproducibility. We also provide one example for each model as a reference.

Results and discussion

Pipeline overview

Briefly, the single-cell analysis pipeline with DANCE platform includes data collection, data downloading, data processing (preprocessing and graph construction), and model development on specific downstream tasks (Fig. 1a).

Benchmark dataset collection

We first collect standard and popular benchmark datasets for each supported downstream task in DANCE. Then, those datasets are organized and cached by dataset name on the cloud.

Data downloading

For each task, DANCE provides a generic interface to load datasets. Since all benchmark datasets supported by DANCE are cached on the cloud in advance, users do not have to download their interested datasets manually. They just need to specify a dataset name 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'
```

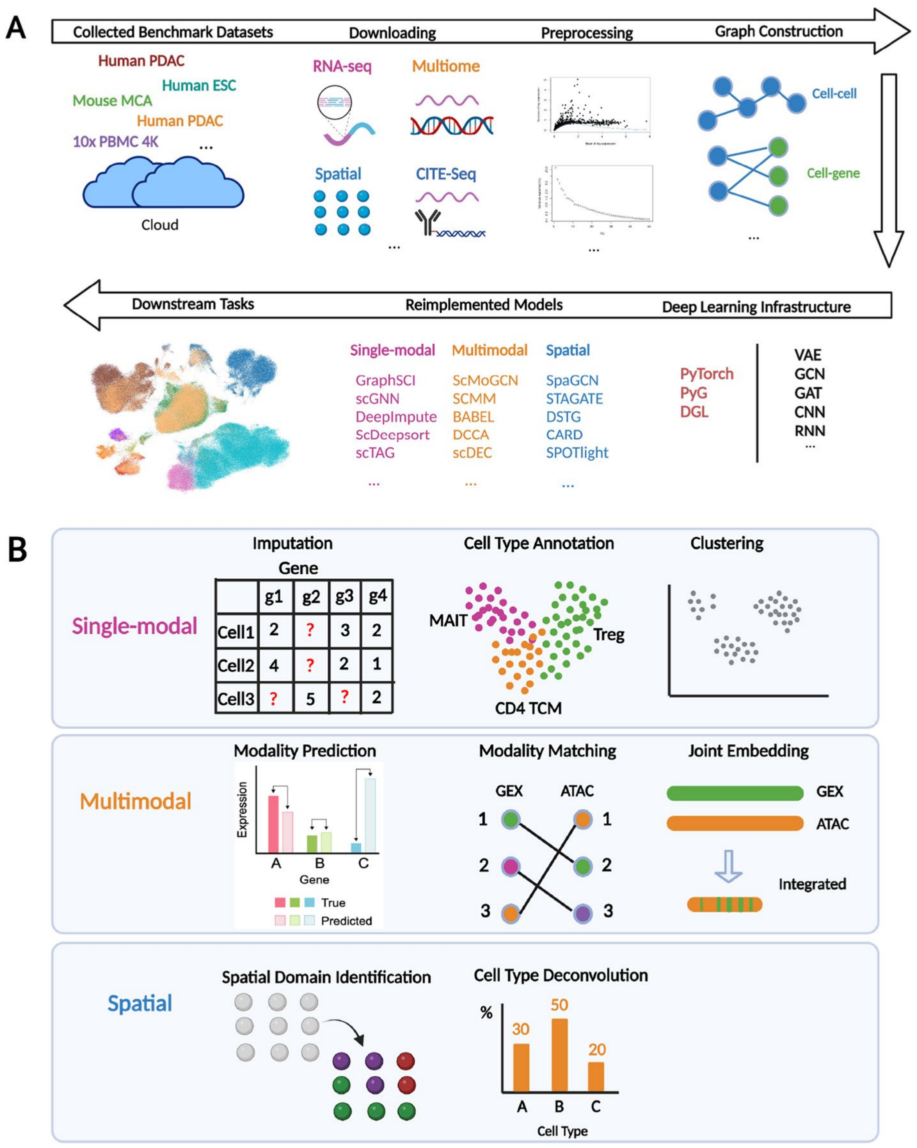

Fig. 1 User perspective of DANCE platform. **a** Overview of single-cell omics analysis pipeline with DANCE platform. Benchmark datasets by the task are organized and cached on the cloud in advance for users' usage. Those data cover scRNA-seq data, multimodal single-cell data like Chromium Single Cell Multiome ATAC + Gene Expression and cellular indexing of transcriptomes and epitopes (CITE-seq), and spatially resolved transcriptomic data. After automatic data downloading from the cloud, the DANCE built-in preprocessing and graph construction (required for graph neural networks model development) are then executed. Subsequently, users can build up their own models via customized deep learning model module in DANCE or utilize our reimplemented state-of-the-art deep learning models in DANCE to accomplish downstream tasks. **b** Currently supported downstream tasks in DANCE

Data processing

After data loading, a collection of data processing methods are 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, and gene filtering. Take graph-sc model as an example, we filter out the rarely expressed genes and normalize the remaining to obtain the same total count for each cell. Then, only the highly expressed genes (top 3000 by default) are kept for clustering [25].

- *Graph construction*: This is required for GNN-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. There are weighted edges between cell nodes and the expressed gene nodes. Let the raw data cell-gene matrix be X, then the weight of gene i to cell j is $w_{ij} = \frac{X[i,j]}{\sum_{k=0}^{m} X[k,j]}$. There is no edge linked between any pairs of cell or gene nodes.

Model development

All types of deep learning models by task have been reimplemented in DANCE with a generic backend framework and unified interface for usage. Users can directly apply them to their interested downstream tasks or build up their own model via our provided customized deep learning module in DANCE.

As shown in Fig. 1b, DANCE presently supports tasks of data spanning through single modality profiling, multimodal profiling, and spatial transcriptomics, which correspond to three stages of single-cell technology development. For a single-modal module, only a single modality like gene expression in the cell can be obtained for analysis. Imputation, cell type annotation, and clustering tasks are supported under this module. For the multimodal module, multiple modalities for the cell can be accessed. For example, CITE-seq can provide both gene expression and protein data for analysis. Modality prediction, modality matching, and joint embedding are currently supported. For the spatial transcriptomics module, the spatial location of the cell in the tissue can be obtained additionally. Spatial domain identification and cell type deconvolution are presently placed under this module. For more details about each task, please refer to the "Methods" section.

Deep learning library

We have seen the rapid development of deep learning in single-cell analysis in recent years [32–39] due to its capability of handling huge, high-dimensional, and sparse data. Among them, GNN, as a branch of deep learning, is playing an increasingly important role in the filed of single-cell analysis [25, 26, 40–44] because it is natural to represent cell-gene in a graph, include prior knowledge into graphs, and extract gene-gene patterns hidden from cell-gene relations via propagation. To facilitate the development of deep learning models in this field, we not only provide all kinds of basic deep learning model implementations like commonly used autoencoders (AEs) [45], generative adversarial networks (GANs) [46], and convolutional neural network (CNN) [47, 48] but also support all types of graph operations like graph convolutional network (GCN) [49] and graph attention network (GAT) [50]. What is more, due to the fact that the original single-cell data is not a graph, we also design several interfaces for users to construct various graphs, like cell-cell, cell-gene, and gene-gene graphs after which one of the GNNs is applied.

Benchmark overview: modules, tasks, models and benchmark datasets

As shown in Fig. 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 GNN-based methods including graph-sc [25], scTAG [51], and scDSC [35] and AE-based methods including scDeepCluster [34] and scDCC [52]. To ensure a systematic evaluation and fair performance comparison of different models, several standard benchmark datasets such as 10X PBMC 4K [53], Mouse Bladder Cells [54], Worm Neuron Cells [55], and Mouse Embryonic Stem Cells [56] 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 the "Methods" section for more details about supported models and datasets.

Comparison with existing packages for single-cell analysis

DANCE is not only acting as a deep learning library to facilitate users' model development but also as a benchmark platform for comprehensive evaluation. 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 [57] proposes an efficient and scalable infrastructure only for spatial omics analysis. DeepCell [58] 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 [59]

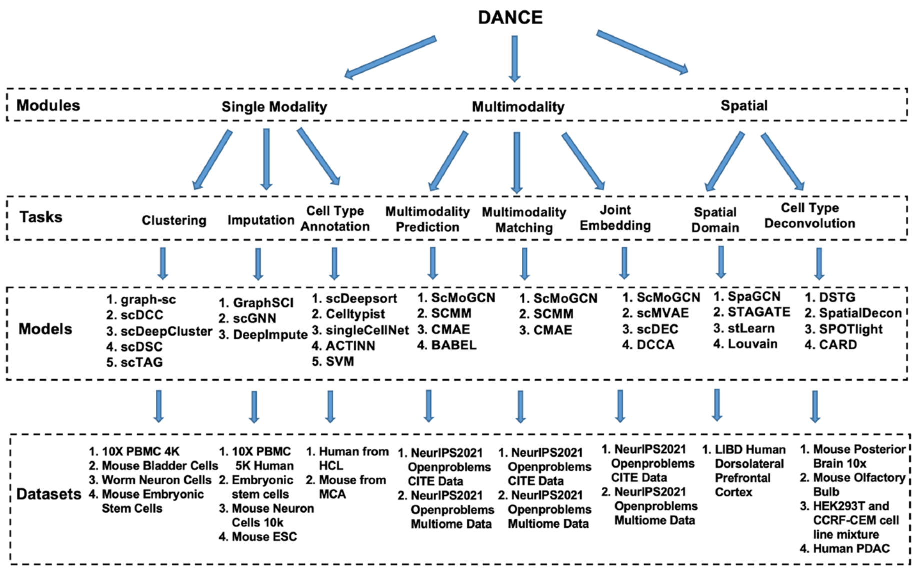
DANCE
Modules
Single Modality
Multimodality
Spatial
Tasks
Clustering
Imputation
Cell Type Annotation
Multimodality Prediction
Multimodality Matching
Joint Embedding
Spatial Domain
Cell Type Deconvolution
Models
1. graph-sc 2. scDCC 3. scDeepCluster 4. scDSC 5. scTAG
1. GraphSCI 2. scGNN 3. DeepImpute
1. scDeepsort 2. Celltypist 3. singleCellNet 4. ACTINN 5. SVM
1. ScMoGCN 2. SCMM 3. CMAE 4. BABEL
1. ScMoGCN 2. SCMM 3. CMAE
1. ScMoGCN 2. scMVAE 3. scDEC 4. DCCA
1. SpaGCN 2. STAGATE 3. stLearn 4. Louvain
1. DSTG 2. SpatialDecon 3. SPOTlight 4. CARD
Datasets
1. 10X PBMC 4K 2. Mouse Bladder Cells 3. Worm Neuron Cells 4. Mouse Embryonic Stem Cells
1. 10X PBMC 5K Human 2. Embryonic stem cells 3. Mouse Neuron Cells 10k 4. Mouse ESC
1. Human from HCL 2. Mouse from MCA
1. NeurIPS2021 Openproblems CITE Data 2. NeurIPS2021 Openproblems Multiome Data
1. NeurIPS2021 Openproblems CITE Data 2. NeurIPS2021 Openproblems Multiome Data
1. NeurIPS2021 Openproblems CITE Data 2. NeurIPS2021 Openproblems Multiome Data
1. LIBD Human Dorsolateral Prefrontal Cortex
1. Mouse Posterior Brain 10x 2. Mouse Olfactory Bulb 3. HEK293T and CCRF-CEM cell line mixture 4. Human PDAC
Fig. 2 A summary of modules, tasks, models, and datasets supported by the DANCE package

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

		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	✗	✗	✗	✗	✗	✓

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 [10] 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 [60, 61] 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 [62] 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 GNN-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 AEs [45] 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.

Unified interface

All models in DANCE are reimplemented in a unified development environment based on python language using Pytorch [29], DGL [30], and PyG [31] as backbone frameworks. What is more, all models in DANCE have generic interfaces for usage. As shown in Fig. 3, data loading is executed in a generic way via dataloader.load_data(), and model.preprocessing_pipeline() works for all datasets and models to specify model specific preprocessing functions. The interfaces of data.get_train_data() and data.get_test_data() are used to get training and test data respectively. For model training and evaluation, the unified interface for model training is model.fit(). 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 scDeepSort [26] for an example, after fitting the model with chosen hyperparameters, we can access the performance of scDeepSort by calling the score function, which will return accuracy to indicate the quality of cell type annotation as a classification task.

Performance showup

To build up a benchmark platform with the capability of systematic evaluations and fair comparisons of available methods, we first collect standard benchmark datasets by task. Then, we reimplement popular existing works for each task in a unified development environment based on the Python programming language with the Pytorch, DGL, and PyG frameworks as the backbone. Finally, we conduct exhaustive experiments of each reimplemented model on collected datasets. The data type supported in DANCE for

```python
# Initialize the model and get the model-specific preprocessing pipeline
model = ScDeepSort()
preprocessing_pipeline = model.preprocessing_pipeline(normalize=args.normalize)

# Load data and perform necessary preprocessing
dataloader = ScDeepSortDataset()
data = dataloader.load_data(transform=preprocessing_pipeline)

# Obtain training and testing data
x_train, y_train = data.get_train_data(return_type="torch")
x_test, y_test = data.get_test_data(return_type="torch")

# Get cell feature graph for scDeepSort
g = data.data.uns["CellFeatureGraph"]
num_genes = data.shape[1]
gene_ids = torch.arange(num_genes)
train_cell_ids = torch.LongTensor(data.train_idx) + num_genes
test_cell_ids = torch.LongTensor(data.test_idx) + num_genes
g_train = g.subgraph(torch.concat((gene_ids, train_cell_ids)))
g_test = g.subgraph(torch.concat((gene_ids, test_cell_ids)))

# Train and evaluate the model
model.fit(g_train, y_train,)
score = model.score(x_test, y_test)
```

Fig. 3 Consistent user experience

benchmarking comes from single modality profiling (RNA, protein, and open chromatin) [1–9], multimodal profiling [10–14], to spatial transcriptomics [15–22]. Currently, DANCE supports three tasks in the single-modality module, three tasks in the multimodality module, and two tasks in the spatial transcriptomics module.

Single-modality module—clustering

Clustering is a key component of single-cell analysis in the single-modality module. Researchers can distinguish between different cell types or cell type subgroups in the gene expression data using clustering. Adjusted Rand Index (ARI) is employed as an evaluation metric. Three GNN-based methods (graph-sc [25], scTAG [51], scDSC [35]) and two AE-based methods (scDeepCluster [34], scDCC [52]) have been reimplemented under this task. scDSC is deep structural clustering for single-cell RNA-seq data (scRNA-seq) using AEs and GNNs in conjunction. graph-sc and scTAG both convert scRNA-seq data to the cell-to-gene graph as an input for the graph encoder, while scTAG takes topology adaptive graph convolutional network (TAGCN) [63] as the graph encoder. scDeepCluster is a ZINB-based AE method for clustering. Similar model structure to scDeepCluster, scDCC additionally adds pairwise constraints into the loss function. Those five reimplemented models are evaluated on our collected four standard benchmarking datasets, which are 10X PBMC 4K [53], Mouse Bladder Cells [54], Worm Neuron Cells [55], and Mouse Embryonic Stem Cells [56]. There are 4271 cells and 16,653 genes with protocol as 10x Genomics in 10X PBMC 4K dataset, 2746 cells and 20,670 genes with protocol as Microwell-seq in Mouse Bladder Cells dataset, 4186 cells and 13,488 genes with protocol as sci-RNA-seq in Worm Neuron Cells dataset, and 2717 cells and 24,175 genes with protocol as Droplet Barcoding in Mouse Embryonic Stem Cells dataset. Figure 4a shows performance comparison between our implementation and the original implementation of five popular methods on 10X PBMC 4K and Mouse Embryonic Stem Cells datasets. We note that our graph-sc implementation increases slightly from 0.7 to 0.709 and from 0.78 to 0.82 on 10X PBMC 4K and Mouse Embryonic Stem Cells datasets respectively. scDCC performs similarly with the original implementation on the first dataset. On the other hand, we can also observe that our scDeepCluster achieves a similar performance to the original one on the first dataset but gets a worse performance on the second dataset since the variance among random seeds on the second dataset is large. What is more, scTAG in the original paper did not report its performance on both datasets. Instead, to have systematic evaluations, we fill the space of all missing reported performance. For the performance of five methods on more datasets, please refer to Additional file 5.

Single-modality module—cell type annotation

In the single-modality module, cell type annotation is to annotate the cell type of an individual cell by comparing the query data to annotated reference data (e.g., a single-cell atlas) or employing marker genes indicative of a particular cell type for annotation or modeling as supervised/semi-supervised learning task. Evaluation of model performance is based on prediction accuracy. Five existing works have been reimplemented under this task. scDeepsort [26] is a pre-trained cell type annotation method developed with a weighted GNN framework. Celltypist [64] is a multinomial logistic regression model

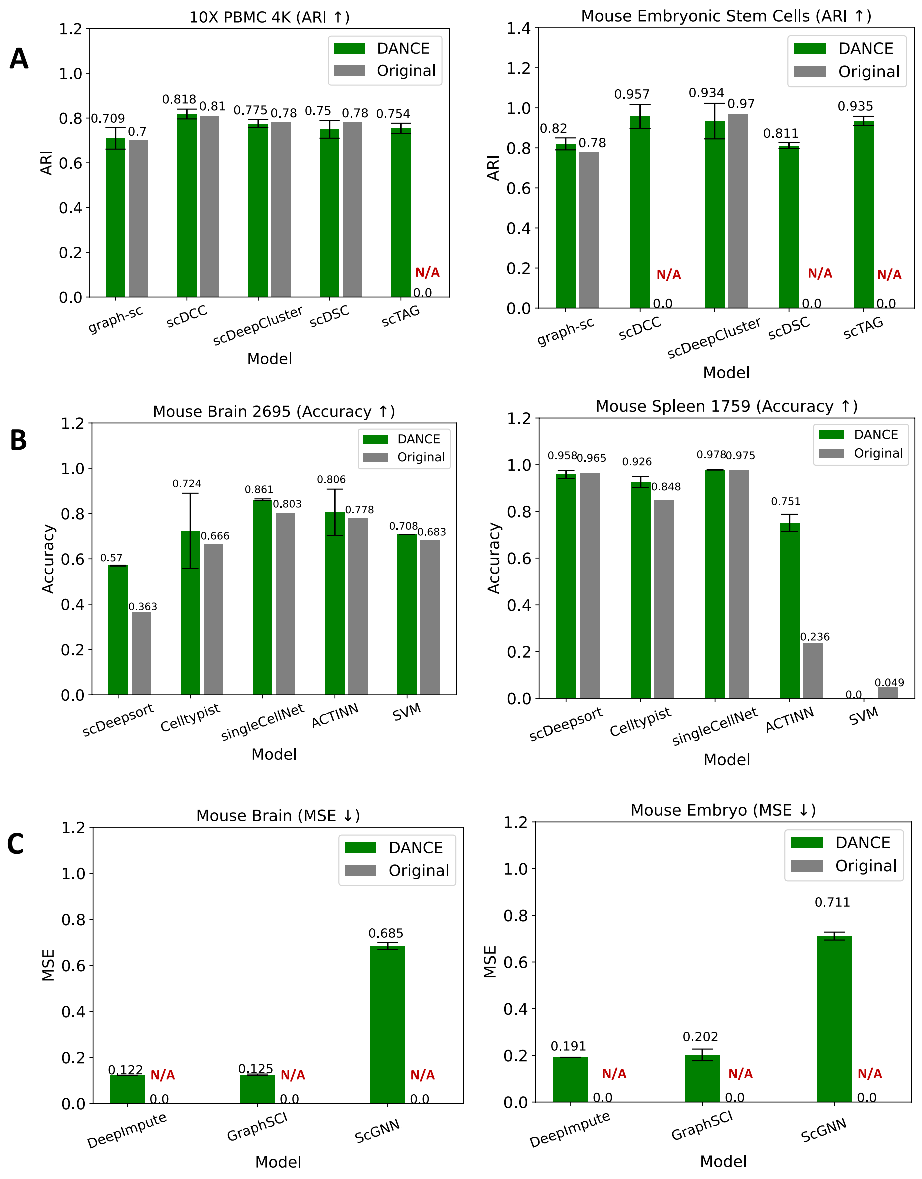

Fig. 4 Performance comparison between our implementation and original implementation for supported tasks in the single-modality module. DANCE result represents the mean performance across 20 randomly chosen seeds, while the original result refers to the performance directly extracted from the original paper. **a** Clustering task. **b** Cell type annotation task. **c** Imputation task. Note: N/A indicates no performance report from the original paper

for classification. SingleCellnet [65] is a random forest-based method, and support vector machine (SVM) [66] is a traditional support vector machine based method to enable the classification of scRNA-seq data. ACTINN [33] is a neural network-based method via multilayer perceptron. Two benchmark datasets have been collected for this task. HCL [67] dataset consists of 562,977 cells, while MCA [68] dataset consists of 201,764 cells. Figure 4b shows performance comparison between our implementation and the original implementation of such five popular methods on the MCA dataset (Mouse Brain 2695 and Mouse Spleen 1759). We can see that most of our implementation models outperform or match the original implementation on both Mouse Brain 2695 and Mouse Spleen 1759 datasets. scDeepsort outperforms the original implementation by a large margin on Mouse Brain 2695 while ACTINN outperforms the original implementation greatly on Mouse Spleen 1759. We also observe that the performance of our scDeepsort and ACTINN is lower than the reported performance from the paper on Mouse Kidney 203 in Additional file 5, which may be explained by the deviation from our implementation or the reported performance from the original paper. For performance comparison on more datasets, please refer to Additional file 5.

Single-modality module—imputation

In the single-modality module, imputation is to correct erroneous zeros by calculating plausible values for gene-cell pairs. For scRNA-seq data, imputation generates false count values for non-expressed genes, but for DNA methylation, imputation provides just the binary one or zero. Mean squared error (MSE) is used as an evaluation metric. Two GNN-based methods and one neural network-based method have been reimplemented under this task. scGNN [40] employs an integrative AE framework that combines gene regulatory signals for scRNA-seq gene expression imputation. GraphSCI [41] employs a graph autoencoder on a cell graph and reconstructs the input using the graph as additional input. DeepImpute [32] constructs multiple neural networks in parallel to infer target genes from an input collection of genes. Four benchmark datasets have been collected for benchmarking under the imputation task. 10X PBMC 5K [69] dataset consists of 5247 cells and 33,570 genes for each cell. Human Embryonic Stem Cells (Human ESC) [70] dataset consists of 758 cells and 17,826 genes for each cell. Mouse Neuron Cells 10k [69] dataset contains 11,843 cells and 31,053 genes for each cell. Mouse ESC [56] dataset is composed of 2717 cells and 24,175 genes for each cell. Figure 4c shows performance comparison on 10X PBMC 5K (Mouse Brain) and Mouse ESC (Mouse Embryo). It is obviously noticed that GNN-based methods like scGNN and GraphSCI outperform simple neural network methods like DeepImpute greatly on both datasets. The performance reports of all three models are missing on both standard benchmark datasets, and we also fill the gap for systematic evaluation.

Multimodality module—modality prediction

In the multimodality module, modality prediction is to predict another modality like antibody-derived tags (ADT) given one modality like single-cell RNA-seq gene expression (GEX) for the same cell. Root mean square deviation (RMSE) is employed to evaluate how well the model performs on the task. Four deep learning-based methods have been reimplemented under this task. scMoGNN [11] for this task is an AE-based method to minimize the loss between one modality and another one. The input to the graph encoder is a cell-feature bipartite graph converted from the original input feature matrix. BABEL [36] trains two neural network-based encoders and two decoders to translate data from one modality to the other and reconstruct itself as well. Cross-modal autoencoders [37] uses AEs to map significantly distinct modalities (including pictures) to a common latent space. scMM [38] is a generative model which makes use of a mixture-of-experts (MoE) multimodal variational autoencoder

(VAE) to investigate the latent dimensions associated with multimodal regulatory programs. Openproblems Neurips2021 competition datasets [71] have been collected for benchmarking under this task. Openproblems Neurips2021 CITE and Openproblems Neurips2021 Multiome contain 81,241 cells and 62,501 cells, respectively. As we can see from Fig. 5a, the GNN-based method scMoGNN outperforms other methods on both sub-task datasets GEX2ADT and GEX2ATAC. GEX2ADT means given GEX to predict ADT while GEX2ATAC means given GEX to predict ATAC. Most of the models have not been tested on those two datasets. We report all performances for a

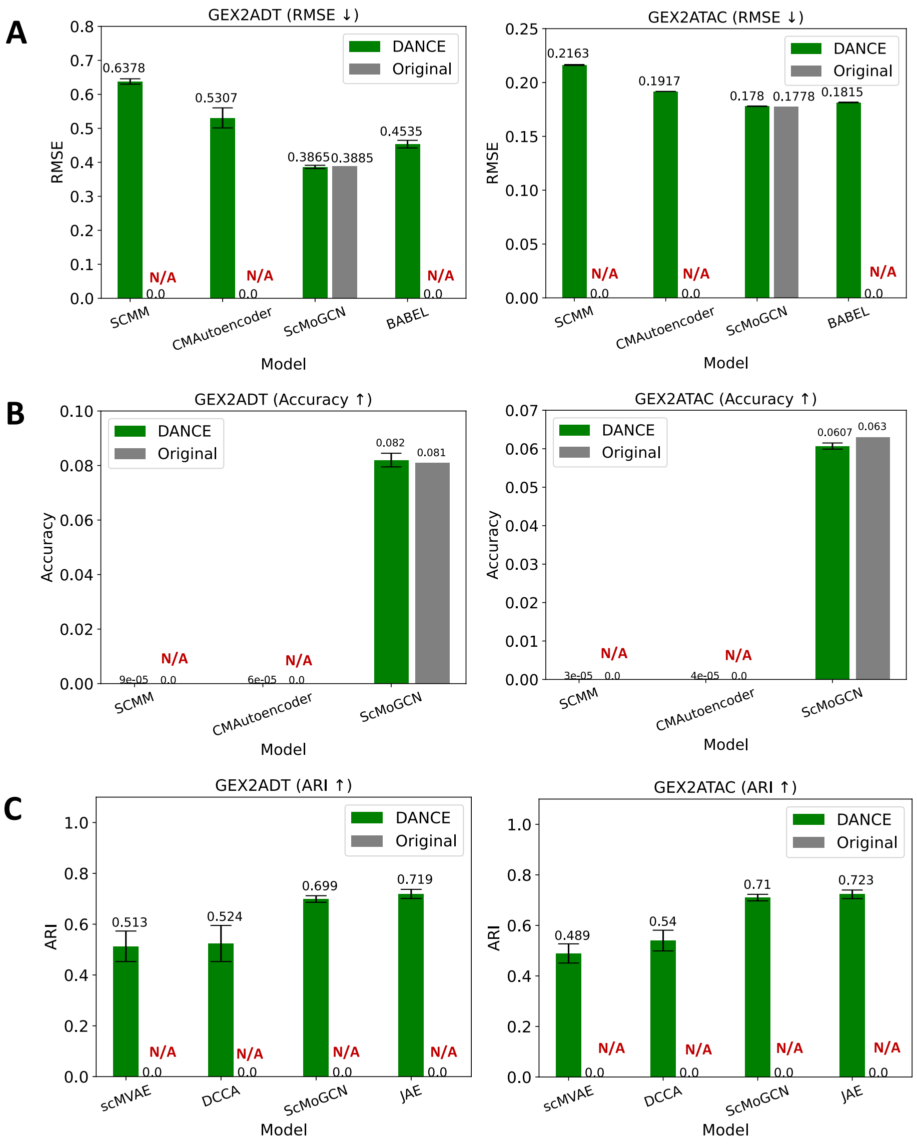

Fig. 5 Performance comparison between our implementation and original implementation for supported tasks in the multi-modality module. DANCE result represents the mean performance across 20 randomly chosen seeds, while the original result refers to the performance directly extracted from the original paper. **a** Modality prediction task. **b** Modality matching task. **c** Joint embedding task. NOTE: N/A indicates no performance report from the original paper

fair comparison. For more sub-task datasets like ADT2GEX and ATAC2GEX, please refer to Additional file 5.

Multimodality module—modality matching

In the multimodality module, modality matching is to match the profiles of each cell from different modalities. The task is evaluated by accuracy. Same with modality prediction, scMoGNN, scMM, and cross-modal autoencoders have been reimplemented with modifications to be suitable for this task. Details of modifications about three models can be found in Additional file 1. The same benchmark datasets with modality prediction have been benchmarked for this task. Similar finding on modality prediction, we can see that GNN-based method scMoGNN outperforms other methods by a huge margin on both sub-task datasets in Fig. 5b.

Multimodality module—joint embedding

In the multimodality module, joint embedding is to learn joint embedding from multiple modalities like GEX and ADT. ARI is employed as an evaluation metric. scMoGNN has been adjusted and implemented to solve this problem. In addition, JAE is a typical AE architecture with an encoder and a decoder, an adapted model from scDEC [72]. scMVAE [39] simultaneously employs three learning methodologies to discover the distribution of multi-omics: product of experts (PoE), neural networks, and concatenation of multi-omics features. DCCA [39] is a VAE-based method. Each VAE undergoes independent training with each modality. Two VAEs are then trained in tandem to optimize the similarity between two latent spaces. Same benchmark datasets with the previous two tasks, as shown in Fig. 5c, GNN-based method scMoGNN resides in TOP 2 on both sub-task datasets. All methods have not been evaluated on both datasets, and we fill the gap for systematic evaluation.

To make the evaluation more systematic and comprehensive, we additional support biology conservation metrics and batch removal metrics under this task in addition to ARI metric. For biology conservation evaluation, normalized mutual information (NMI) metric is adopted to compare the overlap of two clusterings, and cell cycle conservation (Cc_cons) score metric is served as a proxy for the preservation of the signal associated with gene programs during data integration. For batch removal evaluation, ASW batch (ASW_batch) metric is used to quantify batch mixing by taking into account the incompatibility of batch labels per cell type cluster, and graph connectivity (Graph_conn) metric is employed to determine whether cells of the same kind from various batches are embedded close to one another. The benchmarking results for those metrics can be found in Additional file 5.

Spatial transcriptomics module—spatial domain identification

In the spatial transcriptomics module, spatial domain identification seeks to cluster spatial data into a series of meaningful groups. Each group discovered is regarded as a spatial domain. ARI is employed as an evaluation metric for this task. Two GNN-based methods and two traditional methods have been reimplemented. SpaGCN [42] is a GCN-based method for locating geographic domains and variable genes by integrating gene expression and histology. STAGATE [43] is a graph attention method based on AE framework that learns low-dimensional latent embeddings using gene expression and geographical data. Louvain [73] is an iterative strategy for optimizing modularity that is used for network community detection. stLearn [74] does unsupervised clustering on the data that has been normalized by SMEs in order to aggregate similar regions into clusters and find sub-clustering based on the geographic separation of clusters within the tissue. The most popular benchmark dataset LIBD human dorsolateral prefrontal cortex [75] has been collected, and it contains 12 slices. As shown in Fig. 6a, it is obvious that two GNN-based methods SpaGCN and STAGATE outperform others on both slices. Our STAGATE achieves similar results to the original implementation. For average performance on all 12 slices, please refer to Additional file 5.

Spatial transcriptomics module—cell type deconvolution

In the spatial transcriptomics module, cell type deconvolution is to estimate cell type proportions in spatial transcriptomic data. MSE is used as an evaluation metric. One GNN-based method and four traditional methods have been reimplemented under this task. DSTG [44] deconvolutes spatial transcriptomic data using graph-based convolutional networks in order to precisely deconvolve the observed gene expressions at each location and restore their cell constitutions. SPOTlight [76] is a

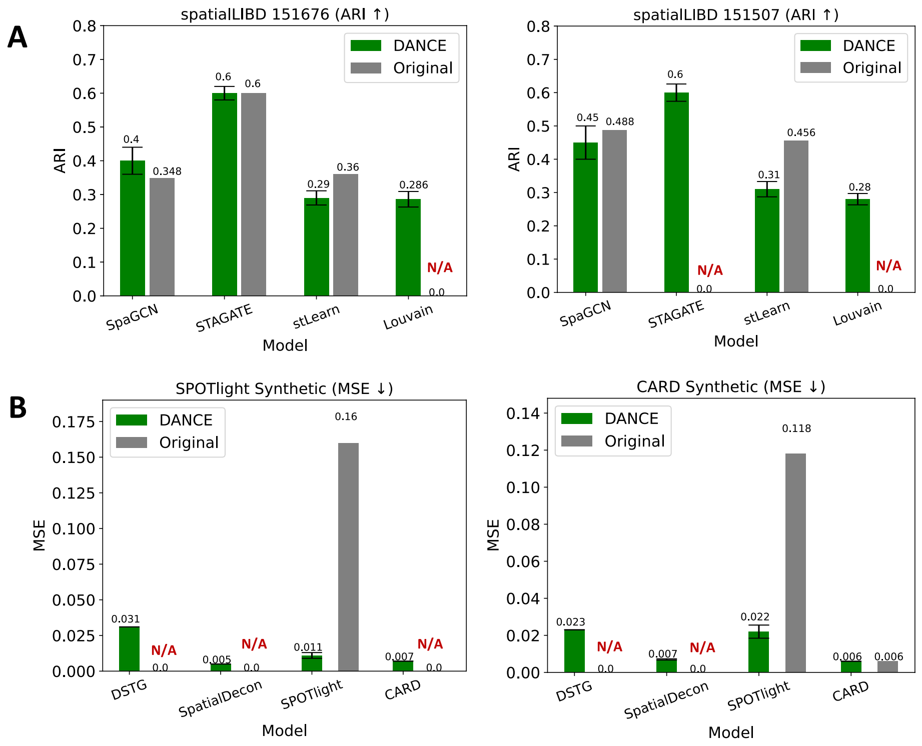

Fig. 6 Performance comparison between our implementation and original implementation for supported tasks in the spatial transcriptomics module. DANCE result represents the mean performance across 20 randomly chosen seeds, while the original result refers to the performance directly extracted from the original paper. **a** Spatial domain identification task. **b** Cell type deconvolution task. Note: N/A indicates no performance report from the original paper

computational method that integrates ST and scRNA-seq data to infer the location of cell types and states within a complicated tissue. It is centered on a seeded non-negative matrix factorization (NMF) regression, which is initialized with cell type marker genes and non-negative least squares (NNLS) to deconvolute ST capture locations (spots). SpatialDecon [77] harnesses log-normal regression and modeling background to quantify cell populations defined by single-cell sequencing within the regions of spatial gene expression. CARD [78] is a conditional autoregressive-based deconvolution that combines cell type-specific expression information from scRNA-seq with correlation in cell type composition across tissue locations. Four standard benchmark datasets have been collected for this task. Mouse Posterior Brain [79] contains 3353 spots, Mouse Olfactory Bulb [80] consists of 1185 spots, HEK293T and CCRF-CEM [81] contain 56 mixtures, and Human PDAC [82] includes 3353 spots. As shown in Fig. 6b, with our reimplementation, SPOTlight improves from 0.16 to 0.011 on Mouse Posterior Brain (SPOTlight Synthetic) and from 0.118 to 0.022 on Mouse Olfactory Bulb (CARD Synthetic). For performance comparison on more datasets, please refer to Additional file 5.

GPU acceleration

In addition to the mentioned similar or better performance of our implementation, we also demonstrate the computational improvement using the GPU implementations in DANCE that are not supported by the original sources. We take SpaGCN and STAGATE models in the task of spatial domain identification as examples. All experiments in this demonstration are carried out on the device with the same memory of 16GB. For GPU supported environment, we use a single node with amd20-v100 while we use a single node with amd20 for CPU running testing. The reported time consumption is from the average of five runs. Following the original papers, SpaGCN is trained with 500 epochs while STAGATE is trained with 2000 epochs. As shown in Fig. 7a, as the number of training cells increases, the computational advantage of our SpaGCN with GPU support becomes more appreciable from the perspective of training time. This phenomenon is more obvious in the STAGATE model as shown in Fig. 7b. The training time

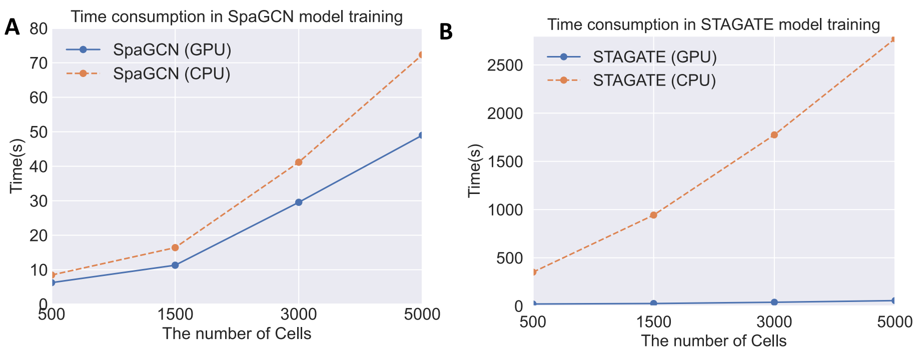

Fig. 7 The comparison of time consumption in model training between implementations with CPU and GPU support. The original implementation comes with CPU but not GPU acceleration while ours is enhanced with GPU support. The SpaGCN and STAGATE with GPU is the implementation in our DANCE. **a** SpaGCN model in spatial domain identification task. **b** STAGATE model in spatial domain identification task

consumption for our STAGATE is 21 s, 26 s, 39 s, and 56 s corresponding to the training cells of 500, 1500, 3000, and 5000 cells, which are 16x, 36x, 45x, and 50x times speedup of the STAGATE implementation with CPU support, respectively. As the number of training cells continues to increase, our acceleration factor will expand greatly.

Easy reproduction

Due to the lack of a publicly available codebase and variances in programming languages, the diversity and complexity of deep learning methods make it difficult for researchers to reproduce the results from the original papers. Another reason specifically for deep learning approaches that cannot be overlooked is hyperparameter tuning. Hyperparameter tuning is to find a set of optimal hyperparameter values for a learning algorithm while applying this optimized algorithm to any data set. This combination of hyperparameters maximizes the performance of the model on a specific dataset. The hyperparameters here are not only model-specific parameters but also the common neural network parameters that must be tuned, such as the number of neurons in the neural network layer, activation function selection, weight decay, and learning rate.

Note that hyperparameter tuning is an empirical task. Based on our many years of tuning experience for deep learning approaches, we execute exhaustive hyperparameter tunning experiments to get optimal model performance on a certain dataset offline and then wrap those optimal hyperparameters values into one command line for reproduction. In such case, users only need to run one command line to obtain reported performance. We have recorded all command lines at the end of model usage example files in DANCE package (https://github.com/OmicsML/dance). The below introduces a few examples of DSTG [44] model on certain benchmark datasets in the task of cell type deconvolution.

DSTG model on CARD synthetic benchmark dataset:

```
$ python dstg.py --dataset CARD_synthetic --nhid 16 --lr
    0.001 --k_filter 50
```

DSTG model on GSE174746 benchmark dataset:

```
$ python dstg.py --dataset GSE174746 --nhid 16 --lr
    0.0001 --k_filter 50
```

DSTG model on SPOTLight synthetic benchmark dataset:

```
$ python dstg.py --dataset SPOTLight_synthetic --nhid 32
    --lr 0.1 --epochs 25
```

In addition, if a newly model is added into DANCE, end users can easily execute the command line by specifying searching space of any interested parameter in the command line for hyperparameter tunning.

Extensible benchmarking

The DANCE platform is the standard, flexible, and extensible benchmark platform for accessing and assessing computational methods across a spectrum of benchmark datasets for a variety of single-cell analysis tasks. Note that even though the limited tasks, models, and benchmark datasets are supported in DANCE, DANCE is being implemented in a highly modular manner, allowing it to be readily expanded and maintained by a community.

We will also keep contributing to include more tasks, models, and benchmark datasets. Our goal is to build up a deep learning and benchmarking community. A HANDS-ON LAB, LIVE TUTORIAL was hosted by the DANCE team in June this year via Zoom to guide users on how to use DANCE including DANCE environment setup, data loading and processing, basic deep learning framework walk-through, example methods providing a detailed, step-by-step tutorial for each task. We have provided a detailed tutorial notebook that can be launched from Google Colab in a dedicated repository (https://github.com/OmicsML/dance-tutorials/blob/tutorial-v1/dance_tutorial.ipynb). We encourage the community to contribute to this extensible benchmark platform to facilitate the overall advancement of single-cell analysis research. For open-source contributions, please refer to Additional file 4 for more details about contribution instructions in DANCE.

Conclusions

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. The performance of our reimplemented models is equivalent or even superior to that of the models in the original papers. We find that the majority of existing works have only reported their performance on limited datasets and in comparison with insufficient methods. We also find that the majority of the existing works cannot consistently perform well through all our collected benchmark datasets. Those prove that comprehensive benchmark datasets and metric evaluation are highly desired in this community. Moreover, we implement all models in a unified development environment based on the python language with Pytorch, DGL, and PyG as backbone frameworks. The interfaces across tasks to download data, read training/test data and train/test models are all unified. Both will greatly facilitate further DANCE development and easy maintenance, and provide a consistent user experience.

Another highlight of DANCE is the easy reproducibility of models. For each task, each implemented method in DANCE is tuned on all gathered standard benchmarks using a grid search to obtain the optimal model, and the corresponding hyperparameters are recorded in a single command line for easy user reproducibility. Last but not the least, due to the nature of extensible feature of our DANCE platform, more additional single-cell analysis tasks, models, and benchmark datasets are easily added and supported into DANCE platform to further enhance the significance and practical utility of the DANCE.

Methods

DANCE implementation and design overview

Environment requirements and setup

DANCE works on python $\geq$ 3.8 and Pytorch $\geq$ 1.11.0. All dependencies are listed in Additional file 2. 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
```

The architecture design and implementation

Figure 8 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.

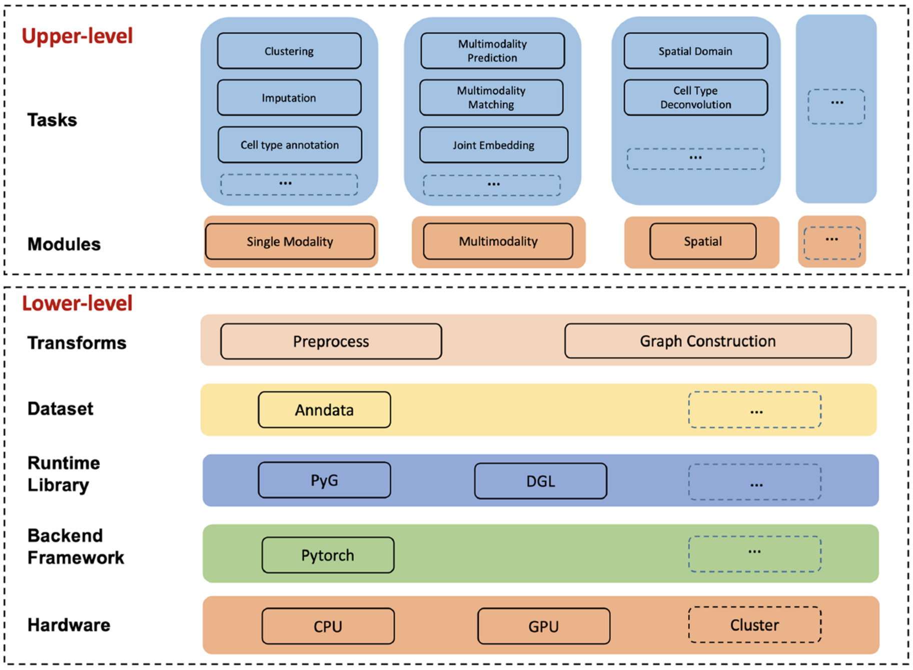

Fig. 8 The architecture of DANCE package

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 [29], 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 DGL [30] and PyG [31] 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 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 is 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 of 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.

Benchmark datasets supported in DANCE

All supported benchmark datasets across 8 tasks in DANCE are summarized in Table 2. 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.

Task definition and evaluation metrics

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 [83]. 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.

Table 2 A summary of all supported benchmark datasets in DANCE

Module	Task	Dataset	Species and tissue	Dataset dimensions	Protocol	Availability
Single modality	**Imputation**	10X PBMC 5K	Human, PBMC	5247 cells 33,570 genes	10x Genomics	[69]
		Human Embryonic Stem Cells (ESC)	Human, ESC	758 cells 17,826 genes	Illumina HiSeq 2500	[70]
		Mouse Neuron Cells 10k	Mouse, Neuron	11,843 cells 31,053 genes	10x Genomics	[69]
		Mouse ESC	Mouse, Neuron	2717 cells 24,175 genes	Droplet Barcod-ing	[56]
	Cell type annotation	HCL	Human	562,977 cells 56 tissues	Smart-seq2	[67]
		MCA	Mouse	201,764 cells 32 tissues	Smart-seq2	[68]
	Clustering	10X PBMC 4K	Human, PBMC	4271 cells 16,653 genes	10x Genomics	[53]
		Mouse Bladder Cells	Mouse, Bladder	2746 cells 20,670 genes	Microwell-seq	[54]
		Worm Neuron Cells	Worm, Nerve	4186 cells 13,488 genes	sci-RNA-seq	[55]
		Mouse Embryonic Stem Cells	Mouse, Embryo	2717 cells 24,175 genes	Droplet Barcod-ing	[56]
Multimodality	**Modality prediction**	Openproblems Neurips2021 CITE	Human, BMMC	81,241 cells 13,953 genes 134 surface proteins	10X TotalSeq B	[71]
		Openproblems Neurips2021 Multiome	Human, BMMC	62,501 cells 13,431 genes 116,490 peaks	10X Multiome	[71]
	Modality matching	Openproblems Neurips2021 CITE	Human, BMMC	81,241 cells 13,953 genes 134 surface proteins	10X TotalSeq B	[71]
		Openproblems Neurips2021 Multiome	Human, BMMC	62,501 cells 13,431 genes 116,490 peaks	10X Multiome	[71]
	Joint embed-ding	Openproblems Neurips2021 CITE	Human, BMMC	81,241 cells 13,953 genes 134 surface proteins	10X TotalSeq B	[71]
		Openproblems Neurips2021 Multiome	Human, BMMC	62,501 cells 13,431 genes 116,490 peaks	10X Multiome	[71]
Spatial	**Spatial domain**	LIBD Human Dor-solateral Prefrontal Cortex	Human, Dorso-lateral prefrontal cortex	12 slices Slice 151673: 3639 spots 33,538 genes	10X Visium	[75]
	Cell type deconvolution	Mouse Posterior Brain	Mouse, Posterior brain	3353 spots 31,053 genes	10X Visium	[79]
		Mouse Olfactory Bulb	Mouse, Olfactory bulb	1185 spots 11,176 genes	10X Visium	[80]

Table 2 (continued)

Module	Task	Dataset	Species and tissue	Dataset dimensions	Protocol	Availability
		HEK293T and CCRF-CEM	Human	56 mixtures 1414 genes	NanoString GeoMx	[81]
		Human PDAC	Human, Pancreas	3353 spots 31,053 genes	Spatial Transcriptomics	[82]

Single-modality module—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.

Single-modality module—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 AE as the backbone. The clustering performance is evaluated by ARI.

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 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.

Multimodality module—modality matching

The objective of the modality matching task is to identify 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, and $\mathbf{S}_{i,j}$ 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 GNNs.

Multimodality module—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 [84], we set the latent dimension size less than or equal to 100. For the evaluation, currently, we only support normalized mutual information(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 task, DANCE now supports 4 models. All of them are deep learning models, one of which is a GNN.

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 [85]. For evaluation, ARI [86] 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.

Spatial transcriptomics module—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. 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. The performance is evaluated by MSE.

Reimplemented models in DANCE

DANCE currently supports total 32 models, which are 3 models in imputation, 5 models in cell type annotation, 5 models in clustering, 4 models in modality prediction, 3 models in modality matching, 4 models in joint embedding, 4 models in the spatial domain, and 4 models in cell type deconvolution. The below will briefly introduce each method. For more details about each model, please refer to Additional file 1.

Single-modality module—imputation

dance.modules.single_modality.imputation.deepimpute

DeepImpute [32] builds multiple neural networks in parallel to impute target genes using a set of input genes.

dance.modules.single_modality.imputation.scgnn

scGNN [40] uses an integrative autoencoder framework for scRNA-seq gene expression imputation that incorporates gene regulatory signals (TRS).

dance.modules.single_modality.imputation.graphsci

GraphSCI [41] is a GNN-based method to impute scRNA-seq data expressions. It uses two autoencoders: one being a graph autoencoder on a cell graph, and the other reconstructs the input using the graph as additional input.

Single-modality module—cell type annotation

dance.modules.single_modality.cell_type_annotation.scdeepsort

Scdeepsort [26] a pre-trained cell type annotation method. It is developed with a weighted GNN framework and then trained on two embedded high-quality scRNA-seq atlases containing 764,741 cells from 88 human and animal tissues.

dance.modules.single_modality.cell_type_annotation.celltypist

Celltypist [64] is a multinomial logistic regression classifier with stochastic gradient descent learning.

dance.modules.single_modality.cell_type_annotation.singlecellnet

SingleCellnet [65] revamped the random forest classifier method to enable the 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 [33] proposes a neural network-based model for cell type annotation. It applies multilayer perceptron for the identification of cell types.

dance.modules.single_modality.cell_type_annotation.svm

SVM is widely adopted as a benchmark in many studies [26, 66]. It works by mapping data to a high-dimensional feature space so that data points can be categorized even when the data are not linearly distinct.

Single-modality module—clustering

dance.modules.single_modality.clustering.scdeepcluster

scDeepCluster [34] is a ZINB-based AE method for clustering.

dance.modules.single_modality.clustering.scdcc

scDCC [52] shares the same model structure as scDeepCluster. In the training process, pairwise constraints are integrated into the loss function.

dance.modules.single_modality.clustering.graphsc

graph-sc [25] is GNN-based method for clustering scRNA-seq data by constructing gene-to-cell graph as the input of graph autoencoder.

dance.modules.single_modality.clustering.sctag

scTAG [51] first generates a K-nearest neighbor cell-to-cell graph. It then adopts a ZINB-based graph autoencoder to process it, which takes TAGCN [63] as the graph encoder.

dance.modules.single_modality.clustering.scdsc

scDSC [35] is deep structural clustering for single-cell RNA-seq data jointly through autoencoder and graph neural network.

Multimodality module—modality prediction

dance.modules.multi_modality.predict_modality.scmogcn scMoGNN [11] is a GNN-based method where the input feature matrix is converted into a cell-feature bipartite graph, where each node represents a cell or feature.

dance.modules.multi_modality.predict_modality.babel

BABEL [36] 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.

dance.modules.multi_modality.predict_modality.cmae

Cross-modal autoencoders [37] use AEs to map vastly different modalities (including images) to a shared latent space.

dance.modules.multi_modality.predict_modality.scmm

scMM [38] leverages a MoE multimodal VAE [87] to explore the latent dimensions that associate with multimodal regulatory programs.

Multimodality module—modality matching

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 for two modalities respectively.

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.

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 for each modality to estimate the variational posterior. In the modality matching task, we hereby take the latent vectors generated by encoders as the source for matching.

Multimodality module—joint embedding

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 reduces the input dimension. Next, the preprocessed features of two modalities are concatenated and jointly considered as feature nodes in the graph construction. scMoGNN is further trained by minimizing a reconstruction loss, a cell type auxiliary loss, and a regularization loss.

dance.modules.multi_modality.joint_embedding.jae

JAE is an adapted model from scDEC [72]. It is proposed by the authors of scDEC in the NeurIPS competition [84] to better leverage cell annotations. Formally, JAE follows the typical AE architecture with an encoder and a decoder.

dance.modules.multi_modality.joint_embedding.scmvae

scMVAE [39] learns the distribution of multi-omics via three learning strategies simultaneously: PoE, neural networks, and concatenation of multi-omics features.

dance.modules.multi_modality.joint_embedding.dcca

In DCCA [39], each modality is modeled by a VAE. Each VAE is first trained separately with each modality. Then, two VAEs are trained together to maximize the similarity between two latent spaces.

Spatial transcriptomics module—spatial domain

dance.modules.spatial.spatial_domain.spagcn

SpaGCN [42] is a GCN-based method via integrating gene expression and histology to find spatial domains and variable genes.

dance.modules.spatial.spatial_domain.stagate

STAGATE [43] is a graph attention-based autoencoder [45] to learn low-dimensional latent embeddings from gene expression and spatial information.

dance.modules.spatial.spatial_domain.louvain

Louvain [73] is an iterative modularity optimization method for network community detection.

dance.modules.spatial.spatial_domain.stlearn

stLearn [74] 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.

Spatial transcriptomics module—cell type deconvolution

dance.modules.spatial.cell_type_deconvo.dstg

DSTG [44] is a GCN-based method whose graph is constructed on mutual nearest neighbors of low-dimensional embeddings of simulated and real mixed-cell data.

dance.modules.spatial.cell_type_deconvo.spotlight

SPOTlight [76] is an extension of NMFReg, with non-negative matrix factorization applied to both the scRNA reference matrix and the mixed-cell expression matrix.

dance.modules.spatial.cell_type_deconvo.spatialdecon

SpatialDecon [77] is a non-negative linear regression-based method that assumes a log-normal multiplicative error model between the mixed-cell data and a cell-profile (signature) matrix.

dance.modules.spatial.cell_type_deconvo.card

CARD [78] applies a conditional autoregressive (CAR) assumption on the coefficients of the classical non-negative linear model between the mixed-cell expression and a cell-profile matrix, constructed from reference scRNA-seq.

Supplementary Information

The online version contains supplementary material available at https://doi.org/10.1186/s13059-024-03211-z.

Additional file 1. Appendix A — Details About Supported Models in DANCE.

Additional file 2. Appendix B — Environment Dependencies in DANCE.

Additional file 3. Appendix C — Codebase Structure Tree in DANCE.

Additional file 4. Appendix D — Contribution Instructions in DANCE.

Additional file 5. Appendix E — More Performance Showup in DANCE.

Additional file 6. Review history.

Authors' contributions

JD conceived the project and led the development of the initial version of the software. RL led the development of the second version. HW, WT, ZL, JV, RS, DM, WJ, WZ, QL, LL, and YW co-developed the software. All authors discussed the results and contributed to the final manuscript with the help of YC, YX, and JT. The authors read and approved the final manuscript.

Review history

The review history is available as Additional file 6.

Peer review information

Andrew Cosgrove was the primary editor of this article and managed its editorial process and peer review in collaboration with the rest of the editorial team.

Funding

This research is supported by the National Institutes of Health (NIH) under grant numbers U01DE029255, U01DE033330, R01HL166508, R01DE026728, RO3DE027399, and the National Science Foundation (NSF) under grant numbers CNS 2246050, IIS1845081, IIS2212032, IIS2212144, IOS2107215, DUE 2234015, DRL 2025244, IOS2035472, and IOS2107215/IIS 2123260, the Army Research Office (ARO) under grant number W911NF-21-1-0198, the Home Depot, Cisco Systems Inc, Amazon Faculty Award, Johnson &Johnson, JP Morgan Faculty Award, and SNAP.

Availability of data and materials

DANCE's open-source code is maintained on GitHub (https://github.com/OmicsML/dance) [88] and published under the BSD2 license. It is also deposited to Zenodo (https://zenodo.org/records/10648047) with assigned DOI: 10.5281/zenodo.10648046 [89]. The online documentation is available at (https://pydance.readthedocs.io/en/latest/) [90]. DANCE is released via the Python packaging index: https://pypi.org/project/pydance/. All collected benchmark datasets used in this study are available from Science Data Bank (https://www.scidb.cn/en/s/nmA7fy#p4) with assigned DOI: 10.1101/2022.10.19.512741 [91].

Declarations

Ethics approval and consent to participate

Not applicable.

Competing interests

The authors declare no competing interests.

Received: 7 February 2023 Accepted: 5 March 2024
Published online: 19 March 2024

References

1. Tang F, Barbacioru C, Wang Y, Nordman E, Lee C, Xu N, et al. mRNA-Seq whole-transcriptome analysis of a single cell. Nat Methods. 2009;6(5):377–82.
2. Kolodziejczyk AA, Kim JK, Svensson V, Marioni JC, Teichmann SA. The technology and biology of single-cell RNA sequencing. Mol Cell. 2015;58(4):610–20.
3. Kashyap V, Sitalaximi T, Chattopadhyay P, Trivedi R. DNA profiling technologies in forensic analysis. Int J Hum Genet. 2004;4(1):11–30.
4. Monckton DG, Jeffreys AJ. DNA profiling. Curr Opin Biotechnol. 1993;4(6):660–4.
5. Panneerchelvam S, Norazmi M. Forensic DNA profiling and database. Malays J Med Sci MJMS. 2003;10(2):20.
6. Li N, Overkleeft HS, Florea BI. Activity-based protein profiling: an enabling technology in chemical biology research. Curr Opin Chem Biol. 2012;16(1–2):227–33.
7. Fujii K, Nakano T, Kawamura T, Usui F, Bando Y, Wang R, et al. Multidimensional protein profiling technology and its application to human plasma proteome. J Proteome Res. 2004;3(4):712–8.
8. Chen EI, Hewel J, Felding-Habermann B, Yates JR. Large scale protein profiling by combination of protein fractionation and multidimensional protein identification technology (MudPIT). Mol Cell Proteomics. 2006;5(1):53–6.
9. Ruoff F, Henes M, Templin M, Enderle M, Bösmüller H, Wallwiener D, et al. Targeted protein profiling of in vivo NIPP-treated tissues using DigiWest technology. Appl Sci. 2021;11(23):11238.
10. Hao Y, Hao S, Andersen-Nissen E, Mauck WM III, Zheng S, Butler A, et al. Integrated analysis of multimodal single-cell data. Cell. 2021;184(13):3573–87.
11. 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. ACM (Association for Computing Machinery); 2022. p. 4153–63. https://dl.acm.org/doi/abs/10.1145/3534678.3539213.
12. Cheow LF, Courtois ET, Tan Y, Viswanathan R, Xing Q, Tan RZ, et al. Single-cell multimodal profiling reveals cellular epigenetic heterogeneity. Nat Methods. 2016;13(10):833–6.
13. Mimitou EP, Lareau CA, Chen KY, Zorzetto-Fernandes AL, Hao Y, Takeshima Y, et al. Scalable, multimodal profiling of chromatin accessibility, gene expression and protein levels in single cells. Nat Biotechnol. 2021;39(10):1246–58.
14. Cadwell CR, Scala F, Li S, Livrizzi G, Shen S, Sandberg R, et al. Multimodal profiling of single-cell morphology, electrophysiology, and gene expression using Patch-seq. Nat Protoc. 2017;12(12):2531–53.
15. Marx V. Method of the Year: spatially resolved transcriptomics. Nat Methods. 2021;18(1):9–14.

16. Wang G, Moffitt JR, Zhuang X. Multiplexed imaging of high-density libraries of RNAs with MERFISH and expansion microscopy. Sci Rep. 2018;8(1):1–13.
17. Chen KH, Boettiger AN, Moffitt JR, Wang S, Zhuang X. Spatially resolved, highly multiplexed RNA profiling in single cells. Science. 2015;348(6233):aaa6090.
18. Crosetto N, Bienko M, Van Oudenaarden A. Spatially resolved transcriptomics and beyond. Nat Rev Genet. 2015;16(1):57–66.
19. Moor AE, Itzkovitz S. Spatial transcriptomics: paving the way for tissue-level systems biology. Curr Opin Biotechnol. 2017;46:126–33.
20. Asp M, Bergenstråhle J, Lundeberg J. Spatially resolved transcriptomes-next generation tools for tissue exploration. BioEssays. 2020;42(10):1900221.
21. Waylen LN, Nim HT, Martelotto LG, Ramialison M. From whole-mount to single-cell spatial assessment of gene expression in 3D. Commun Biol. 2020;3(1):1–11.
22. Teves JM, Won KJ. Mapping cellular coordinates through advances in spatial transcriptomics technology. Mol Cells. 2020;43(7):591.
23. Song Q, Su J, Zhang W. scGCN is a graph convolutional networks algorithm for knowledge transfer in single cell omics. Nat Commun. 2021;12(1):1–11.
24. Wang J, Ma A, Chang Y, Gong J, Jiang Y, Qi R, et al. scGNN is a novel graph neural network framework for single-cell RNA-Seq analyses. Nat Commun. 2021;12(1):1–11.
25. Ciortan M, Defrance M. GNN-based embedding for clustering scRNA-seq data. Bioinformatics. 2022;38(4):1037–44.
26. Shao X, Yang H, Zhuang X, Liao J, Yang P, Cheng J, et al. scDeepSort: a pre-trained cell-type annotation method for single-cell transcriptomics using deep learning with a weighted graph neural network. Nucleic Acids Res. 2021;49(21):e122–e122.
27. Ding J, Venegas J, Lu Q, Wang Y, Wu L, Jin W, et al. SpatialCTD: a large-scale TME spatial transcriptomic dataset to evaluate cell type deconvolution for immuno-oncology. bioRxiv. 2023;2023–04.
28. Molho D, Ding J, Tang W, Li Z, Wen H, Wang Y, et al. Deep learning in single-cell analysis. ACM Trans Intell Syst Technol. 2022. ISSN:2157-6904. EISSN:2157-6912. https://dl.acm.org/doi/10.1145/3641284.
29. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, et al. Pytorch: an imperative style, high-performance deep learning library. Adv Neural Inf Process Syst. 2019;32.
30. Wang M, Zheng D, Ye Z, Gan Q, Li M, Song X, et al. Deep graph library: a graph-centric, highly-performant package for graph neural networks. 2019. arXiv preprint arXiv:190901315.
31. Fey M, Lenssen JE. Fast graph representation learning with PyTorch Geometric. 2019. arXiv preprint arXiv:190302428.
32. Arisdakessian C, Poirion O, Yunits B, Zhu X, Garmire LX. DeepImpute: an accurate, fast, and scalable deep neural network method to impute single-cell RNA-seq data. Genome Biol. 2019;20(1):211.
33. Ma F, Pellegrini M. ACTINN: automated identification of cell types in single cell RNA sequencing. Bioinformatics. 2020;36(2):533–8.
34. Tian T, Wan J, Song Q, Wei Z. Clustering single-cell RNA-seq data with a model-based deep learning approach. Nat Mach Intell. 2019;1(4):191–8.
35. 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. Brief Bioinforma. 2022;23(2):bbac018.
36. Wu KE, Yost KE, Chang HY, Zou J. BABEL enables cross-modality translation between multiomic profiles at single-cell resolution. Proc Natl Acad Sci. 2021;118(15):e2023070118.
37. Yang KD, Belyaeva A, Venkatachalapathy S, Damodaran K, Katcoff A, Radhakrishnan A, et al. Multi-domain translation between single-cell imaging and sequencing data using autoencoders. Nat Commun. 2021;12(1):1–10.
38. 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 Rep Methods. 2021;1(5):100071.
39. Zuo C, Chen L. Deep-joint-learning analysis model of single cell transcriptome and open chromatin accessibility data. Brief Bioinforma. 2021;22(4):bbaa287.
40. Wang J, Ma A, Chang Y, Gong J, Jiang Y, Qi R, et al. scGNN is a novel graph neural network framework for single-cell RNA-Seq analyses. Nat Commun. 2021;12(1):1882.
41. 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. 2021;24(5):102393.
42. Hu J, Li X, Coleman K, Schroeder A, Ma N, Irwin DJ, et al. SpaGCN: Integrating gene expression, spatial location and histology to identify spatial domains and spatially variable genes by graph convolutional network. Nat Methods. 2021;18(11):1342–51.
43. Dong K, Zhang S. Deciphering spatial domains from spatially resolved transcriptomics with an adaptive graph attention auto-encoder. Nat Commun. 2022;13(1):1–12.
44. Song Q, Su J. DSTG: deconvoluting spatial transcriptomics data through graph-based artificial intelligence. Brief Bioinforma. 2021;22(3):1–13.
45. Rumelhart DE, Hinton GE, Williams RJ. Learning representations by back-propagating errors. Nature. 1986;323(6088):533–6.
46. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, et al. Generative adversarial nets. Adv Neural Inf Process Syst. 2014;27.
47. LeCun Y, Bengio Y, et al. Convolutional networks for images, speech, and time series. Handb Brain Theory Neural Netw. 1995;3361(10):1995.
48. O'Shea K, Nash R. An introduction to convolutional neural networks. 2015. arXiv preprint arXiv:151108458.
49. Kipf TN, Welling M. Semi-supervised classification with graph convolutional networks. 2016. arXiv preprint arXiv:160902907.
50. Veličković P, Cucurull G, Casanova A, Romero A, Lio P, Bengio Y. Graph attention networks. 2017. arXiv preprint arXiv:171010903.
51. Yu Z, Lu Y, Wang Y, Tang F, Wong KC, Li X. ZINB-based graph embedding autoencoder for single-cell RNA-seq interpretations. Proc AAAI Conf Artif Intell. 2022;36(4):4671–9.

52. Tian T, Zhang J, Lin X, Wei Z, Hakonarson H. Model-based deep embedding for constrained clustering analysis of single cell RNA-seq data. Nat Commun. 2021;12(1):1–12.
53. Zheng GX, Terry JM, Belgrader P, Ryvkin P, Bent ZW, Wilson R, et al. Massively parallel digital transcriptional profiling of single cells. Nat Commun. 2017;8(1):1–12. Dataset Link: https://support.10xgenomics.com/single-cell-gene-expression/datasets/2.1.0/pbmc4k.
54. Han X, Wang R, Zhou Y, Fei L, Sun H, Lai S, et al. Mapping the mouse cell atlas by microwell-seq. Cell. 2018;172(5):1091–1107. Dataset Link: https://figshare.com/s/865e694ad06d5857db4b.
55. Cao J, Packer JS, Ramani V, Cusanovich DA, Huynh C, Daza R, et al. Comprehensive single-cell transcriptional profiling of a multicellular organism. Science. 2017;357(6352):661–667. Dataset Link: http://atlas.gs.washington.edu/worm-rna/docs/.
56. Klein AM, Mazutis L, Akartuna I, Tallapragada N, Veres A, Li V, et al. Droplet barcoding for single-cell transcriptomics applied to embryonic stem cells. Cell. 2015;161(5):1187–1201. Dataset Link: https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE65525.
57. Palla G, Spitzer H, Klein M, Fischer D, Schaar AC, Kuemmerle LB, et al. Squidpy: a scalable framework for spatial omics analysis. Nat Methods. 2022;19(2):171–8.
58. Van Valen DA, Kudo T, Lane KM, Macklin DN, Quach NT, DeFelice MM, et al. Deep learning automates the quantitative analysis of individual cells in live-cell imaging experiments. PLoS Comput Biol. 2016;12(11):e1005177.
59. Wolf FA, Angerer P, Theis FJ. SCANPY: large-scale single-cell gene expression data analysis. Genome Biol. 2018;19(1):1–5.
60. 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 Biol. 2020;21(1):1–31.
61. Korthauer KD, Chu LF, Newton MA, Li Y, Thomson J, Stewart R, et al. A statistical approach for identifying differential distributions in single-cell RNA-seq experiments. Genome Biol. 2016;17(1):1–15.
62. Gayoso A, Lopez R, Xing G, Boyeau P, Valiollah Pour Amiri V, Hong J, et al. A Python library for probabilistic analysis of single-cell omics data. Nat Biotechnol. 2022;40(2):163–6.
63. Du J, Zhang S, Wu G, Moura JM, Kar S. Topology adaptive graph convolutional networks. 2017. arXiv preprint arXiv:171010370.
64. Domínguez Conde C, Xu C, Jarvis L, Rainbow D, Wells S, Gomes T, et al. Cross-tissue immune cell analysis reveals tissue-specific features in humans. Science. 2022;376(6594):eabl5197.
65. Tan Y, Cahan P. SingleCellNet: a computational tool to classify single cell RNA-Seq data across platforms and across species. Cell Syst. 2019;9(2):207–13.
66. Abdelaal T, Michielsen L, Cats D, Hoogduin D, Mei H, Reinders MJ, et al. A comparison of automatic cell identification methods for single-cell RNA sequencing data. Genome Biol. 2019;20(1):1–19.
67. Han X, Zhou Z, Fei L, Sun H, Wang R, Chen Y, et al. Construction of a human cell landscape at single-cell level. Nature. 2020;581(7808):303–309. https://github.com/ZJUFanLab/scDeepSort/releases/download/Pre/_processed/_data/human/_cell/_atlas.7z.
68. Han X, Wang R, Zhou Y, Fei L, Sun H, Lai S, et al. Mapping the mouse cell atlas by microwell-seq. Cell. 2018;172(5):1091–1107. https://github.com/ZJUFanLab/scDeepSort/releases/download/Pre_processed_data/mouse_cell_atlas.7z.
69. Zheng GXY, Terry JM, Belgrader P, Ryvkin P, Bent ZW, Wilson R, et al. Massively parallel digital transcriptional profiling of single cells. Nat Commun. 2017;8(1):14049. https://doi.org/10.1038/ncomms14049.
70. Chu LF, Leng N, Zhang J, Hou Z, Mamott D, Vereide DT, et al. Single-cell RNA-seq reveals novel regulators of human embryonic stem cell differentiation to definitive endoderm. Genome Biol. 2016;17(1):173. https://doi.org/10.1186/s13059-016-1033-x.
71. Luecken et al. 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. Accessed 29 Aug 2021.
72. Liu Q, Chen S, Jiang R, Wong WH. Simultaneous deep generative modelling and clustering of single-cell genomic data. Nat Mach Intell. 2021;3(6):536–44.
73. Blondel VD, Guillaume J-L, Lambiotte R, Lefebvre E. Fast unfolding of communities in large networks. J Stat Mech: Theory Exp. 2008;2008(10):P10008.
74. Pham D, Tan X, Xu J, Grice LF, Lam PY, Raghubar A, et al. stLearn: integrating spatial location, tissue morphology and gene expression to find cell types, cell-cell interactions and spatial trajectories within undissociated tissues. BioRxiv. 2020:2020–05.
75. Pardo B, Spangler A, Weber LM, Page SC, Hicks SC, Jaffe AE, et al. spatialLIBD: an R/Bioconductor package to visualize spatially-resolved transcriptomics data. BMC Genomics. 2022;23(1):1–5. Dataset Link: http://research.libd.org/spatialLIBD/.
76. 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 Res. 2021;49(9):e50–e50.
77. Danaher P, Kim Y, Nelson B, Griswold M, Yang Z, Piazza E, et al. Advances in mixed cell deconvolution enable quantification of cell types in spatial transcriptomic data. Nat Commun. 2022;13(1):1–13.
78. Ma Y, Zhou X. Spatially informed cell-type deconvolution for spatial transcriptomics. Nat Biotechnol. 2022;40(9):1349–59.
79. Mouse Posterior Brain 10x Visium Data. 2022. https://support.10xgenomics.com/spatial-gene-expression/datasets/1.0.0/V1_Mouse_Brain_Sagittal_Posterior. Accessed 11 July.
80. Mouse Olfactory Bulb Data. 2020. https://www.10xgenomics.com/resources/datasets/adult-mouse-olfactory-bulb-1-standard-1. Accessed 26 Aug.
81. Danaher P, Kim Y, Nelson B, Griswold M, Yang Z, Piazza E, et al. Advances in mixed cell deconvolution enable quantification of cell types in spatial transcriptomic data. Nat Commun. 2022;13(1):1–13. Dataset Link: https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE174746.

82. Moncada R, Barkley D, Wagner F, Chiodin M, Devlin JC, Baron M, et al. Integrating microarray-based spatial transcriptomics and single-cell RNA-seq reveals tissue architecture in pancreatic ductal adenocarcinomas. Nat Biotechnol. 2020;38. https://doi.org/10.1038/s41587-019-0392-8.
83. Lähnemann D, Köster J, Szczurek E, McCarthy DJ, Hicks SC, Robinson MD, et al. Eleven grand challenges in single-cell data science. Genome Biol. 2020;21(1):31. https://doi.org/10.1186/s13059-020-1926-6.
84. Luecken MD, Burkhardt DB, Cannoodt R, Lance C, Agrawal A, Aliee H, 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. https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/158f3069a435b314a80bdcb024f8e422-Abstract-round2.html.
85. Burgess DJ. Spatial transcriptomics coming of age. Nat Rev Genet. 2019;20(6):317.
86. Yeung KY, Ruzzo WL. 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. 2001;17(9):763–74.
87. Shi Y, Paige B, Torr P, et al. Variational mixture-of-experts autoencoders for multi-modal deep generative models. Adv Neural Inf Process Syst. 2019;32.
88. Ding J, Wen H, Tang W, Liu R, Li Z, Venegas J, et al. DANCE: a deep learning library and benchmark platform for single-cell analysis. Github. 2023. https://github.com/OmicsML/dance. Accessed 1 Jan.
89. Ding J, Wen H, Tang W, Liu R, Li Z, Venegas J, et al. DANCE: a deep learning library and benchmark platform for single-cell analysis. Zenodo. 2023. https://zenodo.org/records/10648047. Accessed 15 Feb.
90. Ding J, Wen H, Tang W, Liu R, Li Z, Venegas J, et al. DANCE: a deep learning library and benchmark platform for single-cell analysis. Readthedocs. 2023. https://pydance.readthedocs.io/en/latest/.
91. Benchmark datasets used in DANCE for evaluation. Science Data Bank. 2023. https://www.scidb.cn/en/s/nmA7fy#p4. Accessed 15 Feb.