Bioinformatics, 2021, 1–8
https://doi.org/10.1093/bioinformatics/btab787
Advance Access Publication Date: 19 November 2021
Original Paper

Gene expression

GNN-based embedding for clustering scRNA-seq data

Madalina Ciortan and Matthieu Defrance *

Interuniversity Institute of Bioinformatics in Brussels, Université Libre de Bruxelles, Brussels, Belgium

*To whom correspondence should be addressed.

Associate Editor: Valentina Boeva

Received on June 16, 2021; revised on October 15, 2021; editorial decision on October 26, 2021; accepted on November 15, 2021

Abstract

Motivation: Single-cell RNA sequencing (scRNA-seq) provides transcriptomic profiling for individual cells, allowing researchers to study the heterogeneity of tissues, recognize rare cell identities and discover new cellular subtypes. Clustering analysis is usually used to predict cell class assignments and infer cell identities. However, the high sparsity of scRNA-seq data, accentuated by dropout events generates challenges that have motivated the development of numerous dedicated clustering methods. Nevertheless, there is still no consensus on the best performing method.
Results: *graph-sc* is a new method leveraging a graph autoencoder network to create embeddings for scRNA-seq cell data. While this work analyzes the performance of clustering the embeddings with various clustering algorithms, other downstream tasks can also be performed. A broad experimental study has been performed on both simulated and scRNA-seq datasets. The results indicate that although there is no consistently best method across all the analyzed datasets, *graph-sc* compares favorably to competing techniques across all types of datasets. Furthermore, the proposed method is stable across consecutive runs, robust to input down-sampling, generally insensitive to changes in the network architecture or training parameters and more computationally efficient than other competing methods based on neural networks. Modeling the data as a graph provides increased flexibility to define custom features characterizing the genes, the cells and their interactions. Moreover, external data (e.g. gene network) can easily be integrated into the graph and used seamlessly under the same optimization task.
Availability and implementation: https://github.com/ciortanmadalina/graph-sc.
Contact: matthieu.defrance@ulb.be
Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

The recent progress of single-cell RNA sequencing (scRNA-seq) motivated the research for computational methods to analyze transcriptomic data of individual cells. Because information about sequenced cells is only partial, clustering analysis is usually used to discover cellular subtypes or distinguish and better characterize known ones. Clustering scRNA-seq data introduces several challenges, consisting of dropout events (i.e. false observed zero counts), batch contamination and high dimensionality. As shown in various surveys (Freytag *et al.*, 2017; Kiselev *et al.*, 2019; Menon, 2019; Qi *et al.*, 2020), the scientific community developed numerous approaches to mitigate the computational challenges of scRNA-seq data and produce accurate results. ScRNA (Mieth *et al.*, 2019) uses non-negative matrix factorization to incorporate information from a larger annotated dataset and then applies transfer learning to perform the clustering. CIDR (Clusteing through Imputation and Dimensionlity Reduction) (Lin *et al.*, 2017) performs data imputation before clustering a PCA-reduced (Principal Component Analysis) representation using hierarchical clustering. SOUP (Semisoft Clustering with Pure cells) (Zhu *et al.*, 2019) handles both pure and transitional cells and uses the expression similarity matrix to compute soft cluster memberships. Seurat (Satija *et al.*, 2015) leverages graph-processing techniques like community detection (i.e. the Louvain algorithm) to process the shared nearest neighbor graph and predict cluster assignments. RaceID (Grün *et al.*, 2015) identifies rare cell types and improves clustering performance using K-medoids.

Deep learning was equally used to perform data imputation, embedding learning and clustering. DCA (Deep Count Autoencoder) (Eraslan *et al.*, 2019) proposes a deep count autoencoder to denoise and impute the original data by minimizing a zero-inflated negative binomial (ZINB) loss. This contribution inspired other works such as scDeepCluster (Tian *et al.*, 2019), which added a clustering layer to the DCA model, performing cell cluster assignment after an initial denoising phase. A similar approach to scDeepCluster is followed in the DESC (Deep Embedding Single-cell Clustering) method (Li *et al.*, 2020), which separated the data construction from the clustering phase. scziDesk (Chen *et al.*, 2020) proposes a weighted soft K-means enhancing the scDeepCluster model with a triple loss that factors in the association between similar cells under the same cluster. The proposed loss function

combines the ZINB loss, a weighted soft K-means loss and a KL (Kullback-Leibler) divergence between the Student's t-distribution of the embedding space and that of a target distribution, as proposed in DEC (Deep Embedded Clustering) (Xie *et al.*, 2016). The KL divergence analyzes the pairwise similarity of data points in the latent space and encourages similar points to be clustered in the same cluster. A similar approximation of the ZINB distribution of the expression values is performed by the ScVI (Single-cell Variational Inference) (Lopez *et al.*, 2018) method which, in addition to clustering, provides batch correction, differential expression and visualizations

Recently, graph neural networks (GNNs) have been applied to scRNA-seq data. The common strategies to create the graph processed subsequently with GNNs are the cell-to-cell graph and the gene-to-cell graph, depicted in Supplementary Figure S1. The cell-to-cell graph is used in scGNN (Wang *et al.*, 2021) and represents only the cells as graph nodes. The connections between cells are obtained by training a suite of three multi-modal autoencoders in an iterative way. A feature autoencoder learns an embedding for the cell data, used next for constructing a K-Nearest Neighbor (KNN) graph, corresponding to the cell-to-cell model. The KNN graph is processed with a graph autoencoder to learn a topological embedding for cells and produces cluster assignments. The scGNN model performs both imputation and clustering. The gene-to-cell model was proposed in scDeepSort (Shao *et al.*, 2021). In this case, a weighted graph reproduces closely the information in the expression matrix: both cells and genes are treated as graph nodes, and the gene counts in each cell become the weighted edges between genes and cells. Unlike the cell-to-cell architecture, there are no direct connections between cells, but only between genes and cells. The scDeepSort network is trained in a supervised way (i.e. as a classification problem) using annotated human and mouse data of various tissues. A pretrained model is provided to perform inference on new datasets. However, being supervised, scDeepSort cannot be applied to datasets having different target classes than those present in the original training data, which limits the model's generalization to the diversity of the available input training sets. A detailed presentation of the two graph neural network models is provided in Supplementary Materials.

2 Materials and methods

Given a scRNA-seq matrix $D \in R^{m*n}$ having $\boldsymbol{n}$ samples (i.e. cells) and $\boldsymbol{m}$ features (i.e. transcripts), our method, *graph-sc*, models the expression data as a gene-to-cell graph, processes it with a graph autoencoder network and clusters the resulting cell embeddings with either K-means or Leiden clustering algorithm.

2.1 Preprocessing

Before assembling the graph, the expression matrix is preprocessed with the following steps. All genes expressed in less than two cells are first discarded. The expression level of the remaining genes is then normalized to obtain the same total count for each cell (for each cell, the expression count values are divided by the total count for that cell). After computing the variance across the cells, only the most variable genes (top 3000) are kept in the dataset. The removal of low expressed or less variable genes brings a computational gain in addition to reducing the overall data noise. These operations produce a positive normalized matrix noted $X \in R^{d*n}$, where $\boldsymbol{d}$ is the number of selected features (genes) and $\boldsymbol{n}$ the number of cells. This preprocessing has been chosen as it maximizes the clustering performances, as shown in an ablation study detailed in Supplementary Materials/Ablation studies. A similar preprocessing procedure was also proposed in scziDesk (Chen *et al.*, 2020).

2.2 Graph creation

The scRNA-seq data are modeled as a gene-to-cell graph, bringing several adaptations to the scDeepSort model. The gene-to-cell approach has been selected as a starting point for three theoretical reasons, detailed in Supplementary Materials. First, the gene-to-cell architecture is simple, as it reproduces directly the information in the expression matrix: the gene nodes are connected with the cell nodes in which they are expressed. Second, compared with classical neural-network methods, it provides increased flexibility to model the connections between genes and cells and control how the information is aggregated. Finally, it allows to easily integrate external data into the graph and uses it seamlessly. Other types of omics data, such as the bulk RNA-seq could provide relevant information characterizing the gene co-expression events, which combined with the scRNA-seq data, could help the model produce better cell embeddings. This information can be integrated in the gene-to-cell graph as gene-to-gene connections, refining the local neighborhood of gene nodes. However, this functionality depends on the availability of relevant external data, requirement not always easy to satisfy; as such this track remains an optional improvement to *graph-sc*, presented in Section 4. Next, we detail the steps to create the scRNA-seq gene-to-cell graph. Graph neural networks process graph structures which consist of both an adjacency matrix (describing the connections between nodes, corresponding to the arrows in Fig. 1d) and a node feature matrix, storing representations for each node (corresponding to the vectors next to each node in Fig. 1d). In a nutshell, the node features are essential for the functioning of the graph as all underlying operations combine (e.g. sum) the features of each node with those of neighboring nodes (identified with through the adjacency matrix) to produce hidden representations for each node. Further technical details on the functioning of graph neural networks are provided in Supplementary Materials/Graph Preliminaries. The following section details the creation of graph nodes, graph node features and graph node edges.

Graph nodes. Both genes and cells are created as graph nodes. Each node must have initial values for its features (i.e. a feature vector), which will be processed by a graph neural network input layer. After testing multiple types of graph neural network layers (Supplementary Fig. S9), the convolutional layer (Kipf and Welling, 2017) has been chosen as input layer, as it maximized the clustering performances. This layer processes directly the input graph data. When performing the forward pass, it produces for each node a hidden representation computed as the sum of its own features and those of the neighboring nodes. This operation requires all nodes to be represented in the same input space. Next, the initial feature values for gene and cell nodes are defined. For each gene node, input features representing the top principal components (50) are created using the normalized matrix X. Inspired by the structure of the expression matrix, where a cell is represented by the set of expressed genes, for each cell nodes, initial features are created by summing the corresponding gene nodes. This strategy produces a representation in the same space as the gene nodes (see Fig. 1). Thus, all graph nodes have input features in the same 50D space. The PCA reduction has been chosen for its ability to extract meaningful features from a high-dimensional space. The projection in a lower-dimensional space is a common ground between the cell and the gene nodes and finally, it provides a lower memory footprint than storing for example, the original gene data. The PCA transformation starts by scaling the data such that each gene has zero mean and unit variance. The matrix X has been used instead of the raw data D because normalizing the expression data and scaling the values has shown to produce better representations for scRNA-seq data (Satija *et al.*, 2015) and has been adopted in several state-of-the-art methods (Chen *et al.*, 2020; Tian *et al.*, 2019). Using only the most variable genes, selected in X, by removing the genes with less informative content for clustering provides a gain in the computational speed, as it also limits the graph size. The cell node features are computed as the sum of expressed gene nodes, weighted by the values in the positive matrix X.

Graph edges. Cell nodes are connected to the expressed gene nodes with edges having as weights, the values in the positive matrix X. Thus, the weight of the edge connecting gene ito cell j is $w_{ij} = \frac{D[i,j]}{\sum_{k=0}^{m} D[k,j]}$.

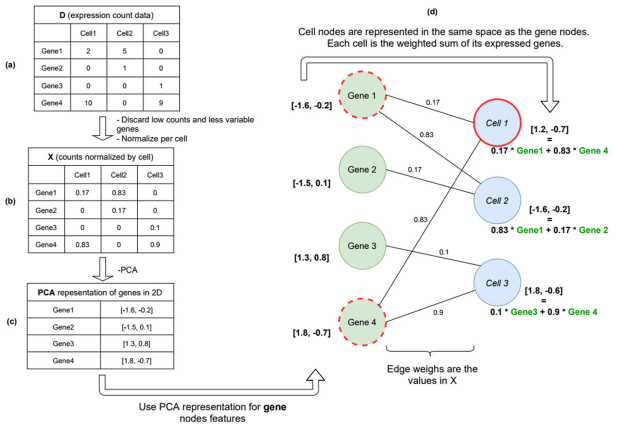

Fig. 1. Overview of graph creation. To model the expression matrix with a graph neural network, a graph defined by node features and an adjacency matrix (containing the weighted edges between nodes) are first produced. A gene-to-cell architecture is adopted to represent both the genes and the cells as graph nodes. Cell nodes are connected to the expressed gene nodes (e.g. Cell 1 is connected to Genes 1 and 4). The preprocessing of the scRNA-seq expression matrix D (**a**) consists of normalizing the data by cells to produce the normalized expression matrix X (**b**). A hidden representation combining the neighboring nodes' features is attributed to each node of the graph, which requires the gene and the cell nodes to be expressed in the same space. The cell nodes are represented as the sum of expressed gene node features. Thus, gene node features are created first and consist of the first principal components (PCs) of the normalized matrix, X. For simplicity, this figure exemplifies the first 2 PCs (**c**). Next, the features for cell nodes are computed as the sum of expressed genes, weighted by X (**d**). In addition, all nodes are enriched with self-connections having a weight of 1

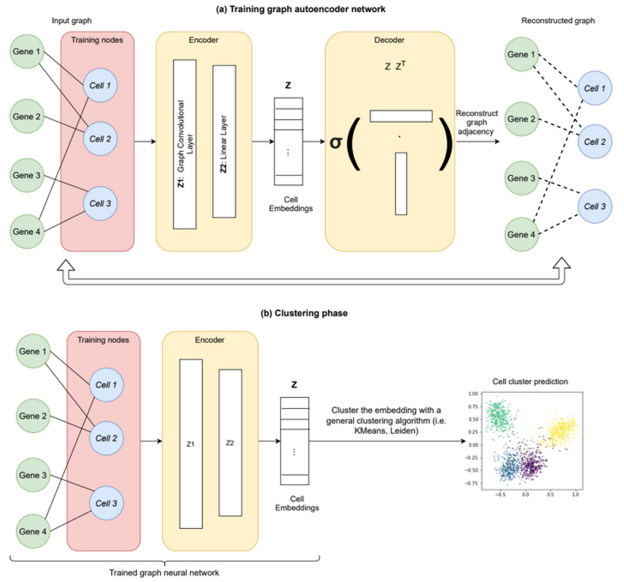

Fig. 2. Method overview. Training phase (**a**). A graph autoencoder neural network is trained to produce hidden representations (embeddings) for the input graph nodes. As the goal is to cluster the cells, the relevant representations are the cell embeddings. Our network consists of an encoder model, which produces node embeddings (Z) and a decoder model, which uses the embeddings (Z) to reconstruct the adjacency matrix and the graph. Our encoder model consists of a Graph Convolutional layer, which aggregates the neighborhood of input nodes as a sum weighted by the input edge values. To produce cell embeddings for clustering, encoder processes only cell nodes (red box). The gene nodes are instrumental in this neighborhood aggregation. Clustering phase (**b**). The result of the convolutional layer is passed through a Linear layer to produce the final cell embeddings, Z. After the network training phase, the produced cell embeddings are clustered to produce cell cluster assignments

Using a weighted graph is an elegant approach to allow genes to have different contributions for different cells, depending on the expression level. The weighted graph provides an alternative to a graph pruning phase (i.e. as proposed in scGNN), which would remove noisy connections based on an arbitrary dataset-dependent threshold. Note that, as depicted in Figure 1, the graph does not include gene-to-gene or cell-to-cell connections and consists only of gene-to-cell connections. Before passing the graph as input to the neural network, the standard edge normalization in graph neural networks is performed, as detailed in Supplementary Materials for scDeepSort. In a nutshell, this edge normalization addresses the gene variability across cells and prevents us from creating aggregations having a different range of values for different cells. All graph nodes are also enriched with self-loop connections with a weight of 1.

2.3 Graph training

As we assume that no knowledge on cell class assignments is available, unlike scDeepSort, we propose an unsupervised model, a convolutional graph autoencoder neural network (Kipf and Welling, 2016), to process the gene-to-cell graph. Autoencoder networks are models trained to reconstruct the input data and in doing so, produce a meaningful representation (i.e. an embedding) of the input data. Graph autoencoders produce representations for input nodes and are trained to reconstruct the adjacency matrix. As illustrated in Figure 2, our autoencoder consists of an encoder and a decoder network. The encoder is composed of a Graph Convolutional Layer, processing the graph (the input node features, Z_0, being the PCA representations) and returning a vectorial representation for all nodes (Z_1). The output of the convolutional layer (Z_1) is passed through a linear layer, producing a new representation of the nodes (Z) which is also the desired representation of cell data. The encoder network iterates only over the cell nodes (highlighted in red in Fig. 2) and produces cell embeddings used subsequently in the clustering phase. However, the gene nodes play an essential role in the neighborhood aggregation of cell nodes performed by the graph convolutional layer.

The second part of the network, the decoder, is trained to reconstruct the graph adjacency matrix from the inner product of the cell embedding $\sigma(Z \cdot Z^T) \sim A$, where σ is the sigmoid function. To keep this presentation simple, the technical details concerning the graph convolutional layer and the training objective is provided in Supplementary Materials.

2.4 Clustering phase

After having produced cell embeddings, a general clustering algorithm is used to produce cell-cluster assignments. The decoupling between the embedding creation and the cluster assignment provides flexibility to adapt to both cases when the expected number of clusters is known (K-means) and unknown (Leiden), but also to analyze the embedding with any other suitable technique.

3 Results

The performances of clustering methods are evaluated with both external scores (assessing the agreement with the provided ground truth): Adjusted Rand Index (ARI) score (Hubert and Arabie, 1985), Normalized Mutual Information and internal scores (assessing the cluster compactness): Silhouette score (Rousseeuw, 1987) and Calinski Harabasz (Caliński and Harabasz, 1974). For all metrics the higher the value, the better the performance. The implementation details together with a detailed description of the evaluation framework are provided in Supplementary Materials, Evaluation Framework section.

3.1 Competing methods

Clustering analysis is typically performed on unlabeled data, either in an explorative way (to discover the number of clusters best fitting the analyzed data) or in an exploitation setting, using prior knowledge or a good definition of the sample groups to be identified. Some existing methods require to input the number of clusters to be

identified, while others, more suitable for exploratory analysis, can dynamically infer it from various data density or connectivity criteria. Our experimental setup compared the performance of *graph-sc* with 12 competing methods, representative of both scenarios. SczíDesk (Chen *et al.*, 2020), scDeepClustering (Tian *et al.*, 2019), scRNA (Mieth *et al.*, 2019), cidr (Lin *et al.*, 2017) and soup (Zhu *et al.*, 2019) take as input the expected number of clusters while scGNN (Wang *et al.*, 2021), Seurat (Satija *et al.*, 2015), scanpy (Wolf *et al.*, 2018) implementation), desc (Li *et al.*, 2020), scedar (Zhang *et al.*, 2020), raceid (Muraro *et al.*, 2016) and scvi (Lopez *et al.*, 2018) perform clustering without any alternative information. In addition, 6 naive baselines (depicted in gray in all our plots) consisting of clustering with K-means the following dimensionality reduced version of the expression matrix were assessed: the first 2 (labeled pca2_kmeans) and 50 (labelled pca50_kmeans) principal components of X, the first 20 (umap20_kmeans) or 50 (umap50_kmeans) UMAP, the first 2 UMAP components of the 50 PCA (pca50_umap_kmeans) of X and with Leiden the best performing baseline, the 2 UMAP components of the 50 PCA of X (labelled pca50_umap_leiden). A detailed record of all benchmarked methods, their repositories and instructions on how to reproduce our experimental results has been made available in Supplementary Table S1.

All our experiments present the results of three consecutive runs of each method on each dataset. The methods annotated with (asterisks) are those that did not receive as input the number of clusters. Our methods are usually highlighted in bold. This experimental setting is used to benchmark the presented methods on a collection of 24 simulated and 15 real scRNA-seq datasets, as detailed below.

3.2 Analysis of simulated data

The data simulation strategy consists of generating datasets approximating various biological scenarios in which we controlled the number of clusters, samples, genes and dropout rates. The R package splatter (Zappia *et al.*, 2017) was used to create balanced and imbalanced datasets (i.e. uniform and non-uniform distribution of cluster sizes). As shown in Figure 3, our methods (*graph-sc* with K-means and Leiden) provide encouraging results, on both external and internal quality measures, reporting high average ARI scores as well as a good ranking (in general, the first or the second position). There is no significant performance difference between K-means and Leiden clustering. The description of the data simulation process and a detailed analysis are provided in Supplementary Materials. As simulated datasets remain an approximation of the biological data, the following sections of the article focus on the analysis of a collection of 15 real-world datasets.

3.3 Analysis of single-cell datasets

A collection of 15 real-world scRNA-seq datasets has been assembled by combining the data made available by scziDesk and scDeepCluster. The datasets from scziDesk have been created at Stanford University from mouse cells using Smart-seq2 and 10x Genomics sequencing (Schaum *et al.*, 2018). The Smart-seq2 datasets have been prefixed with 'Quake Smart' while the latter with 'Quake10x'. Other publicly available datasets have been added, as follows: Adam *et al.* (2017), Muraro *et al.* (2016), Romanov *et al.* (2017) and Young *et al.* (2018). The scDeepCluster data had been collected using four sequencing platforms: 10x genomics platform for the PBMC cells (Zheng *et al.*, 2017), droplet barcoding for mouse embryonic stem cells (Klein *et al.*, 2015), Microwell-seq for mouse bladder cells (Han *et al.*, 2018) and sci-RNA-seq for worm neuron cells (Cao *et al.*, 2017). As detailed in Supplementary Table S4, all datasets are class-imbalanced and contain 4–16 annotated clusters and 870–9552 cells. More details about data sparsity and

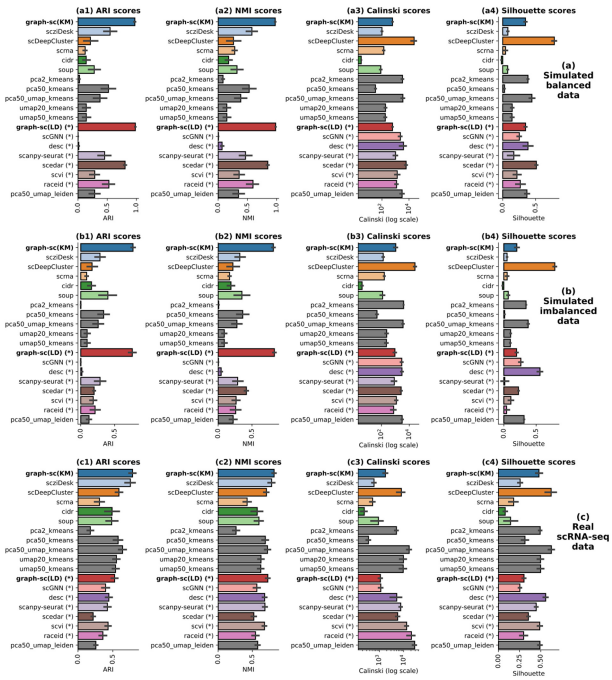

Fig. 3. Method evaluation scores on all simulated balanced (a1–a4), imbalanced (b1–b4) and real-world scRNA-seq (c1–c4) datasets. The ARI scores are depicted in panels 1, normalized mutual information scores in panels 2, Calinski Harabasz scores in panels 3 and Silhouette scores in panels 4. The internal quality scores (Calinski, Silhouette) measure the compactness of the identified clusters in the created cell embedding space. However, those metrics are independent from the external quality scores (i.e. the most compact clusters do not necessarily correspond to the ground truth annotations and conversely). *graph-sc* identifies clusters in agreement with the annotations and having also a good internal quality despite not being the most compact partitions

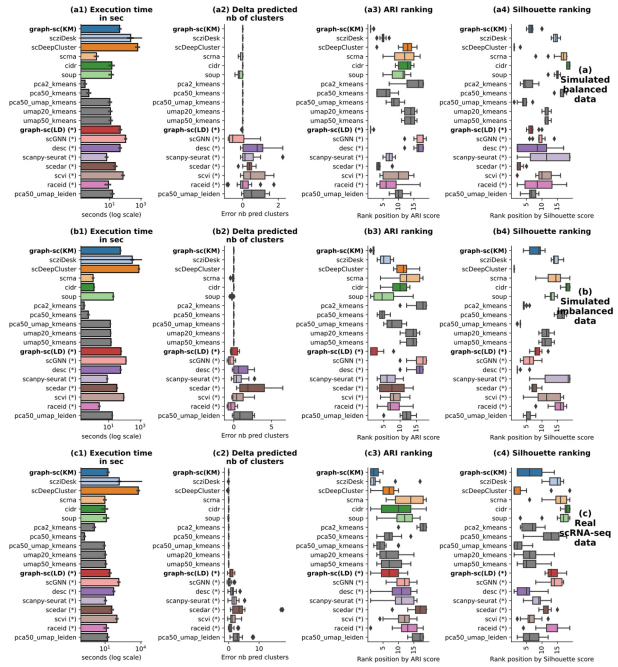

Fig. 4. Method result ranking on all simulated balanced (a1–a4), imbalanced (b1–b4) and real-world scRNA-seq (c1–c4) datasets. The execution time in seconds is depicted in panels 1. The error as the relative difference between the predicted and the true number of clusters [(pred—true)/true] is illustrated in panels 2. A perfect clustering has 0 error, negative values represent methods underestimating the number of clusters in the data and positive scores conversely. The dataset ranking for each method by ARI scores is presented in panels 3 and for Silhouette scores in panels 4. The ranking values vary between 1 and 14, the lower the better. Most methods in the asterisk category tend to overestimate the number of clusters in the data, behavior which is more pronounced in the imbalanced setting

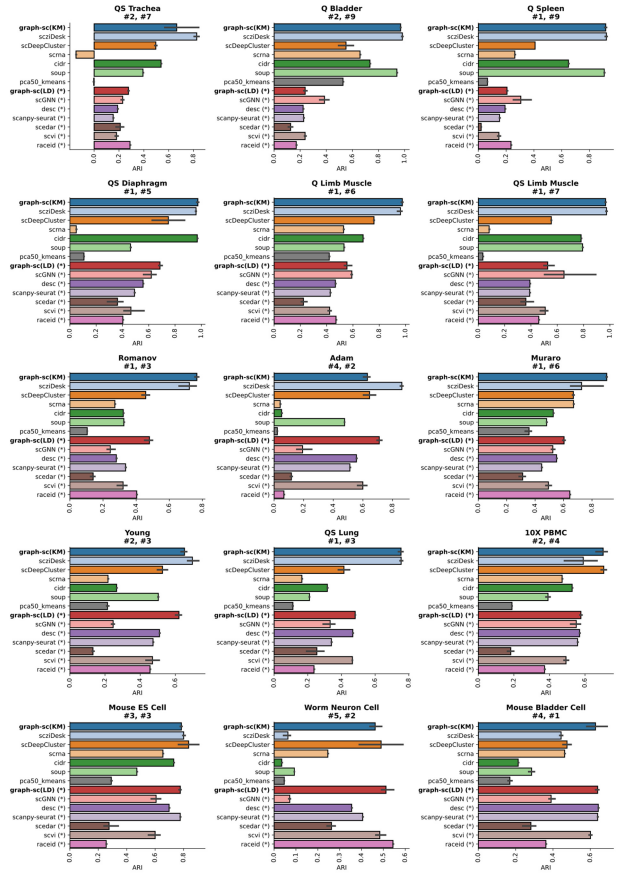

Fig. 5. Dataset-level analysis of real scRNA-seq data on ARI scores. The dataset annotations (e.g. #1) indicate the ranking of *graph-sc*, respectively, with K-means and Leiden clustering on each analyzed dataset

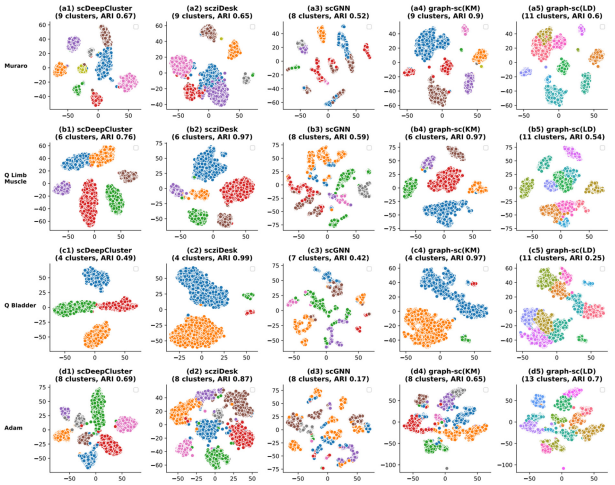

Fig. 6. Visualization of identified clusters. The partitions identified with scDeepCluster, scziDesk, scGNN, *graph-sc*(KM), *graph-sc*(LD) on four datasets (Muraro, Quake Limb Muscle, Quake Bladder, Adam). The remaining datasets are depicted in Supplementary Figures S12 and S13. The plots illustrate the t-SNE 2D projections of the created embeddings. All selected methods start by producing an embedding for the cells, which is clustered in a second phase. The quality of the method depends on both the created embedding and the clustering algorithm. Both our methods clustered the same embedding, produced by *graph-sc*

other descriptive statistics specific to each dataset have been provided in Supplementary Table S5.

The results depicted in Figure 3(c1–c4) indicate that the proposed methods (*graph-sc* with K-means and Leiden) compares favorably with state-of-the-art techniques on real-world datasets, producing average ARI scores of 0.78 and 0.53, respectively. Ran with default parameters, Leiden method overestimates the number of clusters in the data on average by a factor of two, which penalizes the external quality scores and explains the difference in performance compared with K-means. Supplementary Table S9 indicates that when performing hyperparameter optimization on the Leiden's worst performing datasets, our model achieves results comparable with K-means clustering (Figs 4 and 5).

The detailed dataset-level ranking analysis depicted in Figure 6 indicates no consensus regarding the best method across all datasets. *graph-sc* compared favorably with the best competitive techniques, being ranked first on 6 datasets and second on another 5 datasets over the 15 real-world scRNA-seq datasets analyzed. The other best-performing methods are scziDesk, scDeepCluster, cidr and soup. ScGNN has a mixed tendency to overestimate and underestimate the number of clusters in different datasets. While the worst-performing method is pca2_kmeans, pca50_kmeans and pca50_umap_kmeans provided results comparable with other state-of-the-art methods. This can be explained as the first 2 principal components are not enough to capture all relevant variations in the data. The other methods (annotated with asterisks) have a significant tendency to overestimate the number of clusters in the data, on average by a factor of 2 (desc, scanpy-seurat, scvi, raceid), but up to five times (scedar). However, the identified partitions have generally higher internal quality scores, as indicated by both the Silhouette (Fig. 3(c3)) and Calinski (Fig. 3(c4)) scores. This behavior may be attenuated with an additional work of method-specific hyperparameter tuning for each dataset, but this introduces additional computational load and requires defining an experimental setup adapted for each technique, going beyond the scope of a broad benchmarking exercise. For a fair comparison, the same parameters across all experiments were used.

The methods providing the most compact partitions (scDeepCluster, desc, scvi) do not always provide results aligned with the ground truth. The average Silhouette score of 0.48 reported by our best method, *graph-sc* with K-means, suggests that the identified partitions are not only in agreement with the ground truth, but they also consist of well-separated clusters. This finding is also supported by a visual analysis of the reported results. A set of four real-world datasets has been selected for visualization. First, the embeddings and the clusters predicted by *graph-sc* are compared with those created by the competitive methods scziDesk, scDeepCluster and scGNN (Fig. 6). Next, the sample clusters created by *graph-sc* are compared with the ground truth (Fig. 7). A comprehensive analysis of these results is provided in Supplementary Materials. This visualization exercise demonstrates that both the embedding and the cell clusters computed with *graph-sc* are aligned with the provided class annotations while forming well-defined clusters.

3.4 Execution time analysis

A detailed analysis of the execution time and computational complexity is provided in Supplementary Materials. Compared with the studied competing methods, *graph-sc* reports average execution times. However, it is faster than the other techniques based on neural networks.

3.5 Method stability

The stability of *graph-sc* has been studied over consecutive runs and when using only a fraction of input cells (input down-sampling). The detailed analysis provided in Supplementary Materials suggests that our method is generally stable and robust to input down-sampling, being able to provide competitive results even when 25% of the input cells are provided as input.

3.6 Ablation studies

A wide range of ablation studies has been performed to assess the importance of all choices proposed for the input preprocessing,

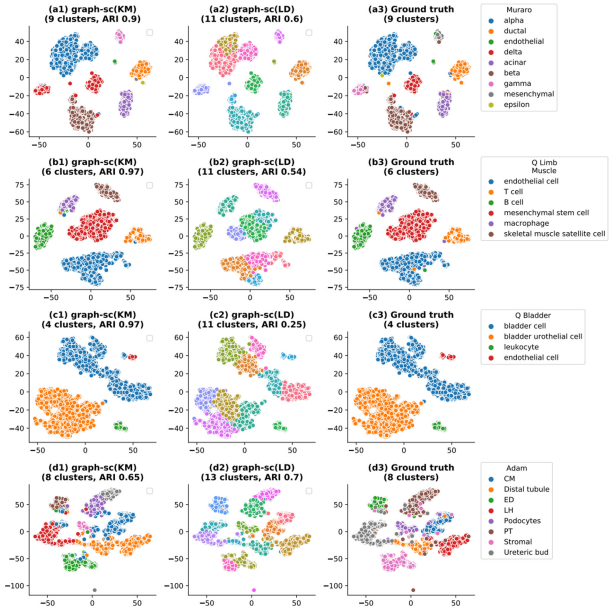

Fig. 7. *graph-sc* clustering results compared with ground truth. The comparison of predicted and ground truth clusters on four scRNA-seq datasets (Muraro, Quake Limb Muscle, Quake Bladder, Adam). Our methods cluster the same embedding, produced with the graph autoencoder. The ground truth is also depicted in the same space. All plots present a 2D t-SNE projection of the underlying embeddings. *graph-sc* (LD) consistently overestimated the number of clusters in the data and performed best on datasets with a large number of clusters. A similar exercise has been performed on simulated data in Supplementary Figures S14–S17

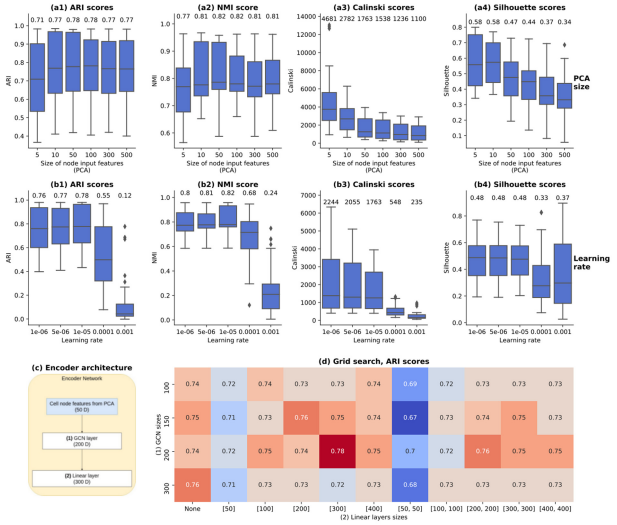

Fig. 8. Identification of optimal parameters. Panels a1–a4 depict the impact on performance of selecting from 5 to 500 principal components from X. Learning rates ranging from 1e-6 to 1e-3 have been explored in panels b1–b4. Panel c reminds the network architecture, consisting of a Graph Convolutional Network (GCN) followed by a linear layer. Forty-four different neural network architectures are explored panel d: on the *y* axis, the explored values for the GCN layer size and on the *x* axis the hidden layer architectures. [300] represents a single linear layer of 300 values while [300, 300] represents 2 stacked linear layers of 300 values each, 'None' indicates no linear layer used in the encoder, in which case the cell embedding is the output of the GCN layer. All experiments are performed three times on all real-world datasets using K-means clustering and we also cross validated the results on three folds (Supplementary Fig. S10)

graph creation, network architecture and training hyperparameters. All default parameters used in our method (i.e. number of PCA components, network architecture, embedding size) are the result of an extensive hyperparameter search exercise, summarized in Figure 8 and, for simplicity, detailed in Supplementary Materials.

4 Discussion

The presented experimental results suggest that *graph-sc* compares favorably with competing methods for clustering scRNA-seq data on simulated and scRNA-seq datasets, achieving encouraging results on both internal and external clustering evaluators. Even though there is no one best method across all analyzed datasets, *graph-sc* ranks the first on 6 out of 15 real-world datasets, when compared with another 12 competing methods. The encouraging results suggest that modeling scRNA-seq data using the proposed gene-to-cell model is an alternative to the analytical method of modeling the dropout, if used to acquire robustness for clustering scRNA-seq data, using NB or ZINB autoencoders (Chen *et al.*, 2020; Eraslan *et al.*, 2019; Tian *et al.*, 2019). Moreover, the decoupling between the embedding generation and the clustering phase allows us to easily explore multiple clustering algorithms and choose the most suitable one. K-means provides the best trade-off between clustering performance and execution time when the expected number of clusters is known. The underlying gain in execution time can allow analyzing a larger range of input parameters, thus providing a solution also to the exploration scenario. In Supplementary Materials is presented an exercise where the BIC and AIC scores have been used to infer the optimal number of clusters when analyzing for each dataset from range of nine candidate values. The results presented in Supplementary Figure S11 indicate that similarly good scores are achieved, having average ARI scores of 0.66 and 0.67, respectively, but lower than when using the ground truth. These runs overestimate the number of clusters in the data, behavior which also characterizes the density-based algorithms. Leiden community detection provides the best results from the algorithms not receiving as input the number of clusters.

Our experimental results demonstrate that *graph-sc* is robust to input down-sampling, stable across consecutive runs and generally insensitive to changes in network hyperparameters. Several types of graph layers have been explored and the best results are achieved with convolutional graph neural networks, the same type of layer used in scGNN. *graph-sc* is faster than similar methods leveraging neural networks and has a smaller memory footprint. The computational efficiency is explained by the reduced number of epochs needed for convergence (10 epochs) in combination with the selection of most variable genes, performed in the preprocessing phase, which limits the number of nodes in the graph.

4.1 Enriching the graph with external information

Adopting a graph-based architecture provides the flexibility not only to choose representations for gene and cell nodes, but also to integrate external information. The gene-to-cell graph can be enriched with gene-to-gene connections, extracted from biologically related datasets and representing co-association events documented in other studies. For this exercise, we analyzed the compilation of 17 mouse tissues bulk RNA transcripts, published by Li *et al.* (2017. The gene associations are computed using the Pearson correlation coefficient across all types of tissues. Next, all pairs of genes having an absolute correlation above an arbitrary threshold of 0.5 are selected, to ensure only important associations are integrated. As depicted in Figure 9a, the pairs of correlated genes produce gene-to-gene edges, weighted by the absolute correlation value.

By selecting the mouse datasets providing information about the gene names, we obtained a set of nine datasets. The experimental results presented in Figure 9b suggest that the performance is only marginally different, and the enriched graph is not systematically more accurate than the baseline. Even though the correlation data come from the same organism, the co-association tendencies observed on average in the bulk datasets of unrelated tissues are not necessarily representative for all mouse scRNA-seq datasets. Thus, this technique can also introduce noise as spurious

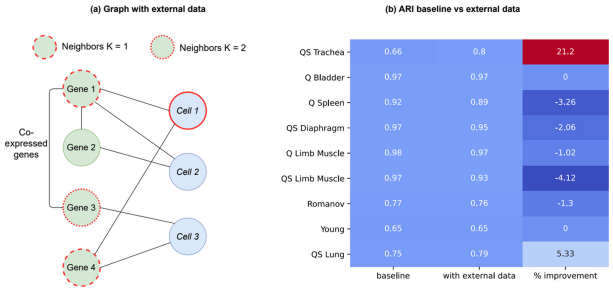

Fig. 9. Enriching the graph with external gene co-association data. In (**a**), external gene correlation data is used to select the pairs of highly correlated genes. Cell 1 expresses Genes 1 and 4 and they contribute to the neighborhood aggregation of Cell 1 at a distance $k = 1$. However, because gene 1 is highly co-expressed with gene 3, an edge is added between Genes 1 and 3 and thus Gene 3 becomes part of the neighborhood aggregation of Cell 1 at a distance $k = 2$. In (**b**) are depicted the ARI scores when using the baseline *graph-sc* in the first column and when *graph-sc* has been enriched with external data in the second column. The third column indicates that when processing the enriched graph, the highest performance gains are achieved (21% and 5% increase in ARI scores), while the remaining runs produce generally low variations, in a range comparable to random initialization variations (Supplementary Fig. S5)

correlations, explaining the variations around the baseline model's score. These results emphasize the importance of selecting the external data to be relevant to the analyzed dataset. Further details on this analysis are provided in Supplementary Materials.

As finding relevant data sources is a requirement not always easy to fulfill, and integrating noisy data can damage the performance, this approach is proposed as an optional improvement to the baseline model. However, integrating multiple external data sources in a single model, under the same optimization problem and with a minimal impact to the overall method is an important theoretical advantage which differentiates our method from the existing approaches.

5 Conclusion

In this article, we proposed a method, *graph-sc*, leveraging graph autoencoders for clustering scRNA-seq data. The proposed method produces competitive results on both simulated and real datasets, it is faster than other similar deep-learning approaches, robust to changes in input parameters and flexible to allow the integration any suitable clustering algorithm. An extensive ablation has been performed, offering insights into the best strategies to create gene-to-cell graphs modeling scRNA-seq data but also into various underlying neural network architectures and training parameters. *graph-sc* can easily incorporate external data with minimal changes to the baseline method, which is another added value compared with existing methods. The external data can be integrated under the same optimization task and we believe that this architectural advantage can open new research directions in combining several types of data to refine the results of existing models. However, this improvement requires the availability of relevant data, a condition which may not always be easy to fulfill. We hope that this work will motivate future research to consider graph models for the analysis of scRNA-seq data.

Data availability

All data needed to reproduce the presented results has been made available on GitHub (https://github.com/ciortanmadalina/graph-sc).

Author contributions

M.C. developed the method, analyzed and interpreted the data. M.C. and M.D. contributed to writing the manuscript. All authors read and approved the final manuscript.

Financial Support: none declared.

Conflict of Interest: The authors declare no competing interests.

References

Adam,M. *et al.* (2017) Psychrophilic proteases dramatically reduce single-cell RNA-seq artifacts: a molecular atlas of kidney development. *Development (Cambridge)*, **144**, 3625–3632.

Caliński,T. and Harabasz,J. (1974) A dendrite method foe cluster analysis. *Commun. Stat.*, **3**, 1–27.

Cao,J. *et al.* (2017) Comprehensive single-cell transcriptional profiling of a multicellular organism. *Science*, **357**, 661–667.

Chen,L. *et al.* (2020) Deep soft K-means clustering with self-training for single-cell RNA sequence data. *NAR Genomics Bioinf.*, **2**, lqaa039.

Eraslan,G. *et al.* (2019) Single-cell RNA-seq denoising using a deep count autoencoder. *Nat. Commun.*, **10**, 1–14.

Freytag,S. *et al.* (2017) Cluster headache: comparing clustering tools for 10x single cell sequencing data. *BioRxiv*, doi:10.1101/203752.

Grün,D. *et al.* (2015) Single-cell messenger RNA sequencing reveals rare intestinal cell types. *Nature*, **525**, 251–255.

Han,X. *et al.* (2018) Mapping the mouse cell atlas by microwell-seq. *Cell*, **172**, 1091–1107.e17.

Hubert,L. and Arabie,P. (1985) Comparing partitions. *J. Classif.*, **2**, 193–218.

Kipf,T.N. and Welling,M. (2016) Variational Graph Auto-Encoders. In: *NIPS Workshop on Bayesian Deep Learning (2016)*, Barcelona, Spain. http://arxiv.org/abs/1611.07308.

Kipf,T.N. and Welling,M. (2017) Semi-supervised classification with graph convolutional networks. In: *5th International Conference on Learning Representations, ICLR 2017 – Conference Track Proceedings*, Toulon, France.

Kiselev,V.Y. *et al.* (2019) Challenges in unsupervised clustering of single-cell RNA-seq data. *Nat. Rev. Genet.*, **20**, 273–282.

Klein,A.M. *et al.* (2015) Droplet barcoding for single-cell transcriptomics applied to embryonic stem cells. *Cell*, **161**, 1187–1201.

Li,B. *et al.* (2017) A comprehensive mouse transcriptomic BodyMap across 17 tissues by RNA-seq. *Sci. Rep.*, **7**, 4200.

Li,X. *et al.* (2020) Deep learning enables accurate clustering with batch effect removal in single-cell RNA-seq analysis. *Nat. Commun.*, **11**, 1–14.

Lin,P. *et al.* (2017) CIDR: ultrafast and accurate clustering through imputation for single-cell RNA-Seq data. *Genome Biol.*, **18**, 59.

Lopez,R. *et al.* (2018) Deep generative modeling for single-cell transcriptomics. *Nat. Methods*, **15**, 1053–1058.

Menon,V. (2019) Clustering single cells: a review of approaches on high-and low-depth single-cell RNA-seq data. *Brief. Funct. Genomics.*, **18**, 434.

Mieth,B. *et al.* (2019) Using transfer learning from prior reference knowledge to improve the clustering of single-cell RNA-Seq data. *Sci. Rep.*, **9**, 20353.

Muraro,M.J. *et al.* (2016) A single-cell transcriptome atlas of the human pancreas. *Cell Syst.*, **3**, 385–394.e3.

Qi,R. *et al.* (2020) Clustering and classification methods for single-cell RNA-sequencing data. *Brief. Bioinf.*, **21**, 1196–1208.

Romanov,R.A. *et al.* (2017) Molecular interrogation of hypothalamic organization reveals distinct dopamine neuronal subtypes. *Nat. Neurosci.*, **20**, 176–188.

Rousseeuw,P.J. (1987) Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.*, **20**, 53–65.

Satija,R. *et al.* (2015) Spatial reconstruction of single-cell gene expression data. *Nat. Biotechnol.*, **33**, 495–502.

Schaum,N. *et al.* (2018) Single-cell transcriptomics of 20 mouse organs creates a Tabula Muris. *Nature*, **562**, 367–372.

Shao,X. *et al.* (2021) scDeepSort: a pre-trained cell-type annotation method for single-cell transcriptomics using deep learning with a weighted graph neural network. *Nucleic Acids Res.*, [Epub ahead of print, doi: 10.1093/nar/gkab775, September 09, 2021].

Tian,T. *et al.* (2019) Clustering single-cell RNA-seq data with a model-based deep learning approach. *Nat. Mach. Intell.*, **1**, 191–198.

Wang,J. *et al.* (2021) scGNN is a novel graph neural network framework for single-cell RNA-Seq analyses. *Nat. Commun.*, **12**, 1882.

Wolf,F.A. *et al.* (2018) SCANPY: large-scale single-cell gene expression data analysis. *Genome Biol.*, **19**, 15.

Xie,J. *et al.* (2016) Unsupervised deep embedding for clustering analysis. In: *33rd International Conference on Machine Learning (ICML 2016)*. Vol. 48, pp. 478–487, New York City, NY, USA.

Young,M.D. *et al.* (2018) Single-cell transcriptomes from human kidneys reveal the cellular identity of renal tumors. *Science*, **361**, 594–599.

Zappia,L. *et al.* (2017) Splatter: simulation of single-cell RNA sequencing data. *Genome Biol.*, **18**, 174.
Zhang,Y. *et al.* (2020) ScEDAR: a scalable Python package for single-cell RNA-seq exploratory data analysis. *PLoS Comput. Biol.*, **16**, e1007794.
Zheng,G.X.Y. *et al.* (2017) Massively parallel digital transcriptional profiling of single cells. *Nat. Commun.*, **8**, 14049.
Zhu,L. *et al.* (2019) Semisoft clustering of single-cell data. *Proc. Natl. Acad. Sci. USA*, **116**, 466–471.