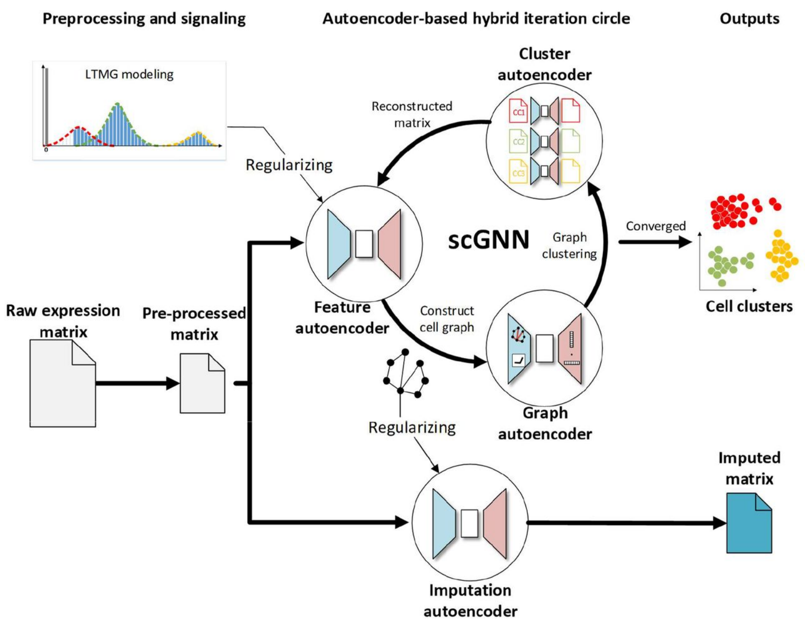
scGNN

Release 0.1

Apr 09, 2021

Main



scGNN (single cell graph neural networks) provides a hypothesis-free deep learning framework for scRNA-Seq analyses. This framework formulates and aggregates cell-cell relationships with graph neural networks and models heterogeneous gene expression patterns using a left-truncated mixture Gaussian model. scGNN integrates three iterative multi-modal autoencoders and outperforms existing tools for gene imputation and cell clustering on four benchmark scRNA-Seq datasets.

CHAPTER 1

scGNN's applications

- Imputed gene expression matrix. to Models heterogeneous gene expression patterns using a left-truncated mixture Gaussian model.
- Learned embedding (features) for clustering.
- Learned graph edges of the cell graph in tuples: nodeA,nodeB,weights.
- Identified cell types. Formulates and agregates cell-cell relationships with graph neural networks

CHAPTER 2

Reference

Wang, J., Ma, A., Chang, Y. et al. scGNN is a novel graph neural network framework for single-cell RNA-Seq analyses. Nat Commun 12, 1882 (2021). https://doi.org/10.1038/s41467-021-22197-x

CHAPTER 3

ACKNOWLEDGEMENTS

This work was supported by the National Institutes of Health's National Institute of General Medical Sciences (awards R35-GM126985 and R01-GM131399).

CHAPTER 4

Support

Feel free to submit an issue or send us an email. Your help to improve scGNN is highly appreciated.

Note: We recommend users to infer LTMG from their datasets. LTMG can improve performance on our benchmarks despite it consumes extra time in data preprocessing. We also provide supports without LTMG to save running time.

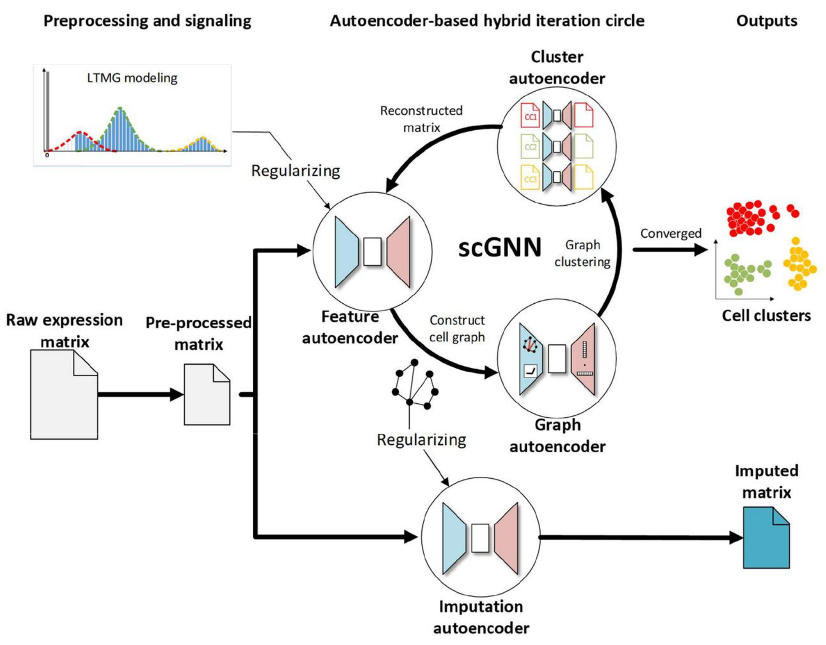
4.1 About

scGNN (single cell graph neural networks) provides a hypothesis-free deep learning framework for scRNA-Seq analyses. This framework formulates and aggregates cell-cell relationships with graph neural networks and models heterogeneous gene expression patterns using a left-truncated mixture Gaussian model. scGNN integrates three iterative multi-modal autoencoders and outperforms existing tools for gene imputation and cell clustering on four benchmark scRNA-Seq datasets. scGNN integrates three iterative multi-modal autoencoders and outperforms existing tools for gene imputation and cell clustering on four benchmark scRNA-Seq datasets. In an Alzheimer's disease study with 13,214 single nuclei from postmortem brain tissues,scGNN successfully illustrated disease-related neural development and the differential mechanism. scGNN provides an effective representation of gene expression and cell-cell relationships. It is also a novel and powerful framework that can be applied to scRNA-Seq analyses

4.1.1 scGNN model

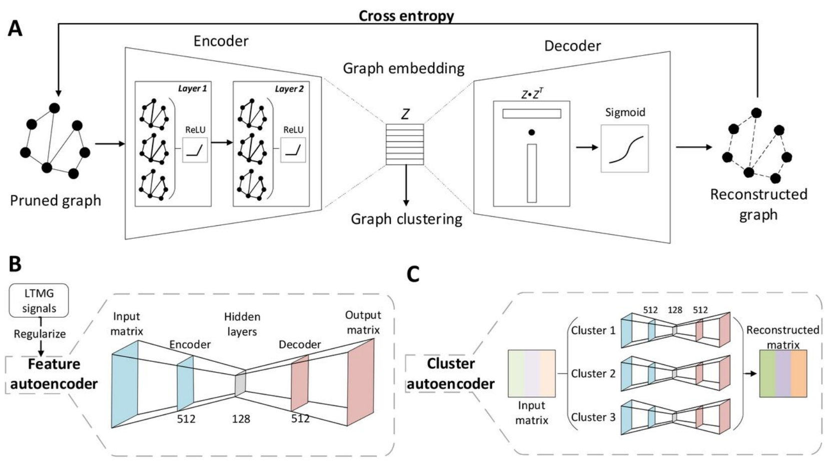
The architecture of scGNN is comprised of stacked autoencoders.

It has three comprehensive computational components in an iteration process, including gene regulation integration in a feature autoencoder, cell graph representation in a graph autoencoder, gene expression updating in a set of parallel cell-type105 specific cluster autoencoders, as well as the final gene expression recovery in an imputation autoencoder.



This is the architecture of scGNN. It takes the gene expression matrix generated from scRNA-Seq as the input. LTMG can translate the input gene expression data into a discretized regulatory signal as the regularizer for the feature autoencoder. The feature autoencoder learns a dimensional representation of the input as embedding, upon which a cell graph is constructed and pruned. The graph autoencoder learns a topological graph embedding of the cell graph, which is used for cell type clustering. The cells in each cell type have an individual cluster autoencoder to reconstruct gene expression values. The framework treats the reconstructed expression as a new input iteratively until converging. Finally, the imputed gene expression values are obtained by the feature autoencoder regularized by the cell-cell relationships in the learned cell graph on the original preprocessed raw expression matrix through the imputation autoencoder

The architecture of scGNN Autoencoders



(A) The graph autoencoder takes the adjacent matrix of the pruned graph as the input. The encoder consists of two layers of GNNs. In each layer, each node of the graph aggregates information from its neighbors. The encoder learns a low dimensional presentation (i.e., graph embedding) of the pruned cell graph. The decoder reconstructs the adjacent matrix of the graph by dot products of the learned graph embedding followed by a sigmoid activation function. The graph autoencoder is trained by minimizing the cross-entropy loss between the input and the reconstructed graph. Cell clusters are obtained by applying k-means and Louvain on the graph embedding. (B) The feature autoencoder takes the expression matrix as the input, regularized by LTMG signals. The dimensions of the encoder and decoder layers are 512×128 and 128×512, respectively.

The feature autoencoder is trained by minimizing the difference between the input matrix and the output matrix. (C) The cluster autoencoder takes a reconstructed expression matrix from the feature autoencoder as the input. An individual encoder is built on the cells in each of the identified clusters, and each autoencoder is trained individually. The concatenation of the results from all clusters is treated as the reconstructed matrix.

4.2 Installation

Installation Tested on Ubuntu 16.04, CentOS 7, MacOS catalina with Python 3.6.8 on one NVIDIA RTX 2080Ti GPU.

4.2.1 From Source:

Start by grabbing this source codes:

```
git clone https://github.com/juexinwang/scGNN.git
cd scGNN
```

4.2.2 Option 1 : (Recommended) Use python virutal environment with condahttps://anaconda.org/

```
conda create -n scgnnEnv python=3.6.8 pip
conda activate scgnnEnv
pip install -r requirements.txt
```

If want to use LTMG (**Recommended** but Optional, will takes extra time in data preprocessing):

```
conda install r-devtools
conda install -c cyz931123 r-scgnnltmg
```

4.2.3 Option 2 : Direct install individually

Need to install R packages first, tested on R >=3.6.1:

In R command line:

```
install.packages("devtools")
library(devtools)
install_github("BMEngineeR/scGNNLTMG")
```

Then install all python packages in bash.

```
pip install -r requirements.txt
```

4.2.4 Option 3: Use Docker

Download and install docker.

Pull docker image **gongjt057/scgnn** from the dockerhub. Please beware that this image is huge for it includes all the environments. To get this docker image as base image type the command as shown in the below:

```
docker pull gongjt057/scgnn:code
```

Type `docker images` to see the list of images you have downloaded on your machine. If **gongjt057/scgnn** in the list, download it successfully.

Run a container from the image.

```
docker run -it gongjt057/scgnn:code /bin/bash
cd /scGNN/scGNN
```

Then you can proceed to the next step.

4.3 Release

4.4 References

Please cite:

Wang, J., Ma, A., Chang, Y. et al. scGNN is a novel graph neural network framework for single-cell RNA-Seq analyses. Nat Commun 12, 1882 (2021). https://doi.org/10.1038/s41467-021-22197-x

The codes are available at https://github.com/juexinwang/scGNN

4.5 Contact

Juexin Wang: wangjue@missouri.edu

Anjun Ma: Anjun.Ma@osumc.edu

Qin Ma: Qin.Ma@osumc.edu

Dong Xu: xudong@missouri.edu

4.6 Getting started

scGNN: a novel graph neural network model for single-cell transcriptome analysis. This is a overview to how to quikly start scGNN.

4.6.1 1. Prepare datasets

4.6.2 2. Data Processing

4.6.3 3. Run scGNN

4.6.4 4. Check Results

4.6.5 5. Downstream Analysis

4.7 Prepare datasets

Datasets should be preprocessed to proceed. scGNN accepts scRNA-seq data format: CSV and 10X

4.7.1 1. Dowmload datasets

CSV format explicitly shows data in plain text. We take an example of Alzheimer's disease datasets (GSE138852) analyzed in the manuscript.

10X format stores scRNA-seq expression sparsely, so is commonly used for huge datasets. Take an example of liver cellular landscape study from human cell atlas(https://data.humancellatlas.org/)

Example:

CSV format

Take an example of Alzheimer's disease datasets (GSE138852) analyzed in the manuscript.

```
mkdir GSE138852
wget -P GSE138852/ https://ftp.ncbi.nlm.nih.gov/geo/series/GSE138nnn/GSE138852/suppl/
↪GSE138852_counts.csv.gz
```

10X format

Take an example of liver cellular landscape study from human cell atlas(https://data.humancellatlas.org/). Click the download link of 'homo_sapiens.mtx.zip' in the page, and get 4d6f6c96-2a83-43d8-8fe1-0f53bffd4674.homo_sapiens.mtx.zip. (It looks like they does not provide direct download link anymore)

```
mkdir liver
wget -P liver https://data.humancellatlas.org/project-assets/project-matrices/
↪4d6f6c96-2a83-43d8-8fe1-0f53bffd4674.homo_sapiens.mtx.zip
cd liver
unzip 4d6f6c96-2a83-43d8-8fe1-0f53bffd4674.homo_sapiens.mtx.zip
cd ..
```

4.8 Data Processing

This step generates Use_expression.csv (preprocessed file) and gets discretized regulatory signals as ltmg.csv from Left-Trunctruncated-Mixed-Gaussian(LTMG) model (Optional but recommended).

In preprocessing, parameters are used:

- **filetype** defines file type (CSV or 10X(default))
- **geneSelectnum** selects a number of most variant genes. The default gene number is 2000
- **inferLTMGTag** (Optional) add –inferLTMGTag to infer LTMG in preprocessing. Need to install r-scgnnltmg. The running time of inferring LTMG is depended on the cell number and gene number selected, i.e. ~10 minutes in GSE138852 and extra ~13 minutes in data liver.

4.8.1 CSV format

Cell/Gene filtering without inferring LTMG:

```
python -W ignore PreprocessingscGNN.py --datasetName GSE138852_counts.csv.gz --
↪datasetDir GSE138852/ --LTMGDir GSE138852/ --filetype CSV --geneSelectnum 2000
```

(Optional) Cell/Gene filtering and inferring LTMG:

```
python -W ignore PreprocessingscGNN.py --datasetName GSE138852_counts.csv.gz --
↪datasetDir GSE138852/ --LTMGDir GSE138852/ --filetype CSV --geneSelectnum 2000 --
↪inferLTMGTag
```

4.8.2 10X format

Cell/Gene filtering without inferring LTMG:

```
python -W ignore PreprocessingscGNN.py --datasetName 481193cb-c021-4e04-b477-
↪0b7cfef4614b.mtx --datasetDir liver/ --LTMGDir liver/ --geneSelectnum 2000 sparseOut
```

(Optional) Cell/Gene filtering and inferring LTMG:

```
python -W ignore PreprocessingscGNN.py --datasetName 481193cb-c021-4e04-b477-
↪0b7cfef4614b.mtx --datasetDir liver/ --LTMGDir liver/ --geneSelectnum 2000 --
↪inferLTMGTag
```

4.9 Run scGNN

Program ***scGNN.py*** is the main entrance of scGNN to impute and clustering. There are quite a few parameters to define to meet users' requirements.

4.9.1 Required

- **datasetName** defines the folder of scRNA-Seq
- **LTMGDir** defines folder of the preprocessed LTMG output
- **outputDir** Output folder of the results

4.9.2 Clustering related

- **clustering-method** Clustering method on identifying celltypes from the embedding. Default is LouvainK: use Louvain to determine the number of the clusters and then use K-means. Supporting clustering type: Louvain/KMeans/SpectralClustering/AffinityPropagation/AgglomerativeClustering/AgglomerativeClusteringK/Birch/BirchN/MeanS
- **n-clusters** predefines the number of clusters, it only used for clustering methods need a number of clusters input as KNN
- **maxClusterNumber** defines the maximum number of cluster allowed, default is 30. This parameter prevents extreme cases that too many clusters identified by Louvian clustering
- **minMemberinCluster** defines the minimum number of cells in a cluster, default is 5. This parameter prevents extreme cases that too many clusters identified by Louvain clustering.
- **resolution** controls the number of clusters identified by Louvain clustering. This parameter can be set between 0.4 and 1.2 in most cases. According to results on benchmarks, we set default 'auto'.

4.9.3 Optional: Hyperparameters

- **EM-iteration** defines the number of iteration, default is 10
- **Regu-epochs** defines epochs in Feature Autoencoder initially, default is 500
- **EM-epochs** defines epochs in Feature Autoencoder in the iteration, default is 200
- **cluster-epochs** defines epochs in the Cluster Autoencoder, default is 200
- **k** is k of the K-Nearest-Neighour Graph
- **knn-distance** distance type of building K-Nearest-Neighour Graph, supported type: euclidean/cosine/correlation (default: euclidean)
- **GAEepochs** Number of epochs to train in Graph Autoencoder

4.9.4 Optional: Performance

- **quickmode** whether or not to bypass the Cluster Autoencoder.
- **useGAEembedding** whether use Graph Autoencoder
- **regulized-type** is the regularized type: noregu/LTMG, default is to use LTMG
- **alphaRegularizePara** alpha in the manuscript, the intensity of the regularizer
- **EMregulized-type** defines the imputation regularizer type:noregu/Graph/Celltype, default: Celltype
- **gammaImputePara** defines the intensity of LTMG regularizer in Imputation
- **graphImputePara** defines the intensity of graph regularizer in Imputation
- **celltypeImputePara** defines the intensity of celltype regularizer in Imputation
- **L1Para** defines the intensity of L1 regularizer, default: 1.0
- **L2Para** defines the intensity of L2 regularizer, defualt: 0.0
- **saveinternal** whether output internal results for debug usage

4.9.5 Optional: Speed

- **no-cuda** defines devices in usage. Default is using GPU, add –no-cuda in command line if you only have CPU.
- **coresUsage** defines how many cores can be used. default: 1. Change this value if you want to use more.

4.9.6 Example:

4.9.7 CSV format

For CSV format, we need add **–nonsparseMode**

Without LTMG:

```
python -W ignore scGNN.py --datasetName GSE138852 --datasetDir ./  --outputDir␣
↪outputdir/ --EM-iteration 2 --Regu-epochs 50 --EM-epochs 20 --quickmode --
↪nonsparseMode
```

(Optional) Using LTMG:

```
python -W ignore scGNN.py --datasetName GSE138852 --datasetDir ./ --LTMGDir ./ --
↪outputDir outputdir/ --EM-iteration 2 --Regu-epochs 50 --EM-epochs 20 --quickmode --
↪nonsparseMode --regulized-type LTMG
```

4.9.8 10X format

Without LTMG:

```
python -W ignore scGNN.py --datasetName 481193cb-c021-4e04-b477-0b7cfef4614b.mtx --
↪datasetDir liver/ --outputDir outputdir/ --EM-iteration 2 --Regu-epochs 50 --EM-
↪epochs 20 --quickmode
```

(Optional) Using LTMG:

```
python -W ignore scGNN.py --datasetName 481193cb-c021-4e04-b477-0b7cfef4614b.mtx --
↪LTMGDir liver/ --datasetDir liver/ --outputDir outputdir/ --EM-iteration 2 --Regu-
↪epochs 50 --EM-epochs 20 --quickmode --regulized-type LTMG
```

On these demo dataset using single cpu, the running time of demo codes is ~33min/26min. User should get exact same results as paper shown with full running time on single cpu for ~6 hours. If user wants to use multiple CPUs, parameter **–coresUsage** can be set as **all** or any number of cores the machine has.

4.10 Check Results

In outputdir now, we have four output files.

- *_**recon.csv**: Imputed gene expression matrix. Row as gene, col as cell. First row as gene name, First col as the cell name.
- *_**embedding.csv**: Learned embedding (features) for clustering. Row as cell, col as embeddings. First row as the embedding names (no means). First col as the cell name.
- *_**graph.csv**: Learned graph edges of the cell graph in tuples: nodeA,nodeB,weights. First row as the name.
- *_**results.txt**: Identified cell types. First row as the name.

For a complete list of options provided by scGNN:

```
python scGNN.py --help
```

4.11 Downstream Analysis

scGNN can effectively impute scRNA-Seq data and accurately predict cell clusters.To assess the imputation and cell clustering performance of scGNN, four scRNA datasets (i.e., Chung, Kolodziejczy, Klein, and Zeisel) with gold-standard cell type labels are chosen as the benchmarks.

4.11.1 Imputation performance of scGNN

The median L1 distance between the original dataset and the imputed values for these corrupted entries were evaluated to compare scGNN with MAGIC, SAUCIE, SAVER, scImpute, scVI, DCA, and DeepImpute.

(A) The L1 distance (the lower the better) and cosine similarity (the higher the better) comparing a 10% leave-out test between scGNN and seven imputation tools on the Klein and Zeisel datasets. scGNN achieved the best scores in both datasets, indicating its superior performance in gene expression recovery (Figure 3A).

(B) Co-expression patterns can be addressed more explicitly after applying scGNN on the Klein data. No clear gene pair relationship of Ccnd3 versus Pou5f1 (upper panel) and Nanog versus Trim28 (lower panel) is observed in the raw data (left) compared to the observation of unambiguous correlations within each cell type after scGNN imputation (right) (Figure 3B).

(C) Comparison of DEG logFC scores using the original expression value (x-axis) and the scGNN imputed expression values (y-axis) identified in Day 1 cells of the Klein data (up) and Microglia cells of the Zeisel data (bottom). The differentiation signals are amplified after imputation (Figure 3C).

(D) The relationships of four more gene pairs are also enhanced. Comparison of gene co-expression relationships in the Klein dataset.Different colors indicate cell clusters given in the original paper (Day 1, 2, 7, and 9) (Figure S3).

(E) In the Zeisel dataset, scGNN amplifies differentially expressed genes (DEGs) signals with a higher fold change than the original, using an imputed matrix to confidently depict the cluster heterogeneity (Figure 3C and Figure S4).

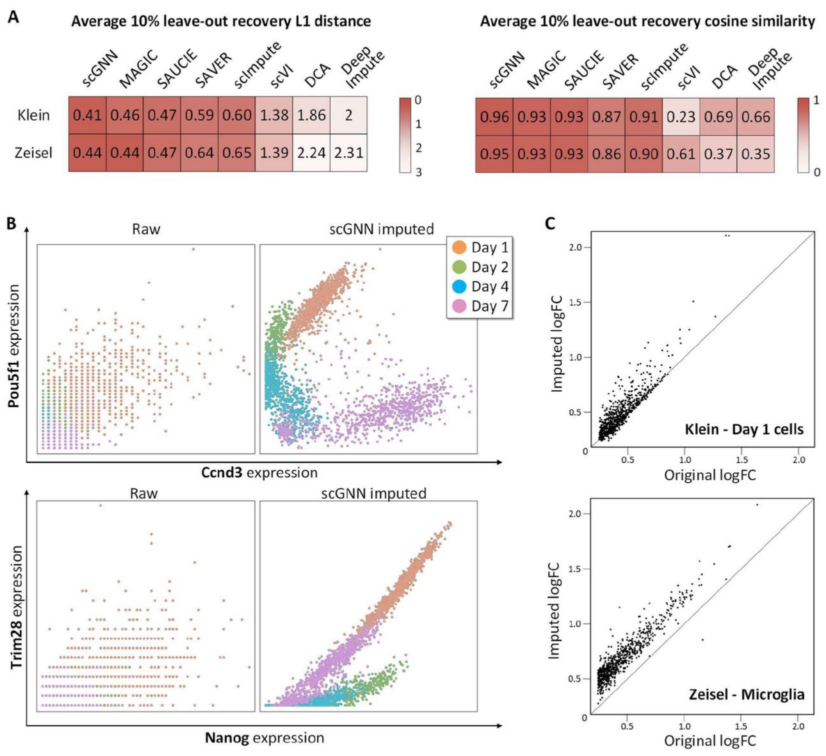


Figure 3. Comparison of the imputation performance.

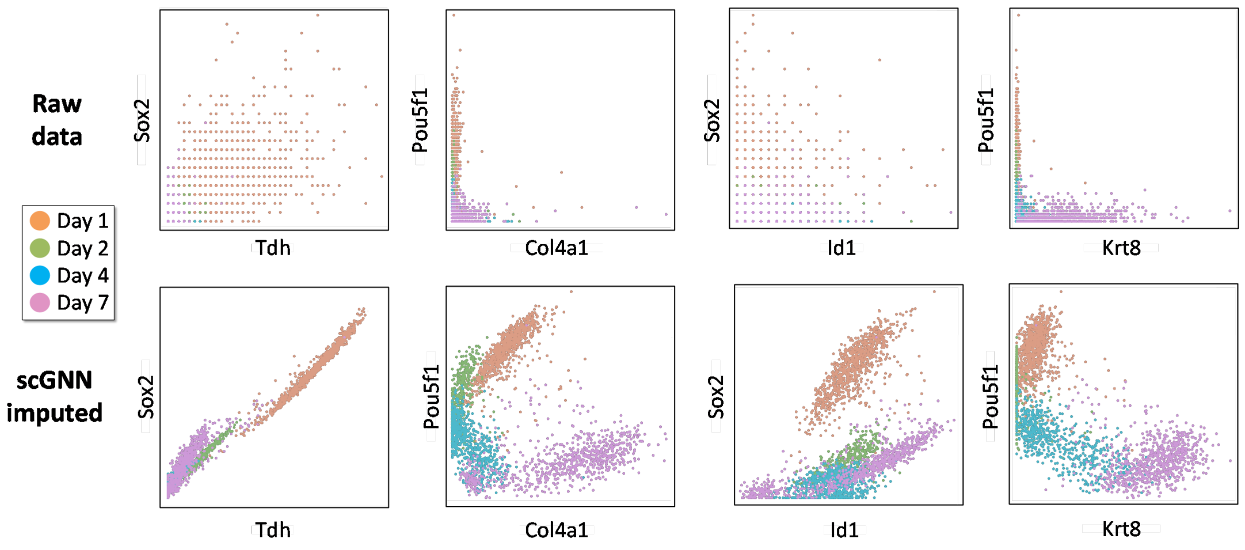


Figure S3. Comparison of gene co-expression relationships in the Klein dataset.

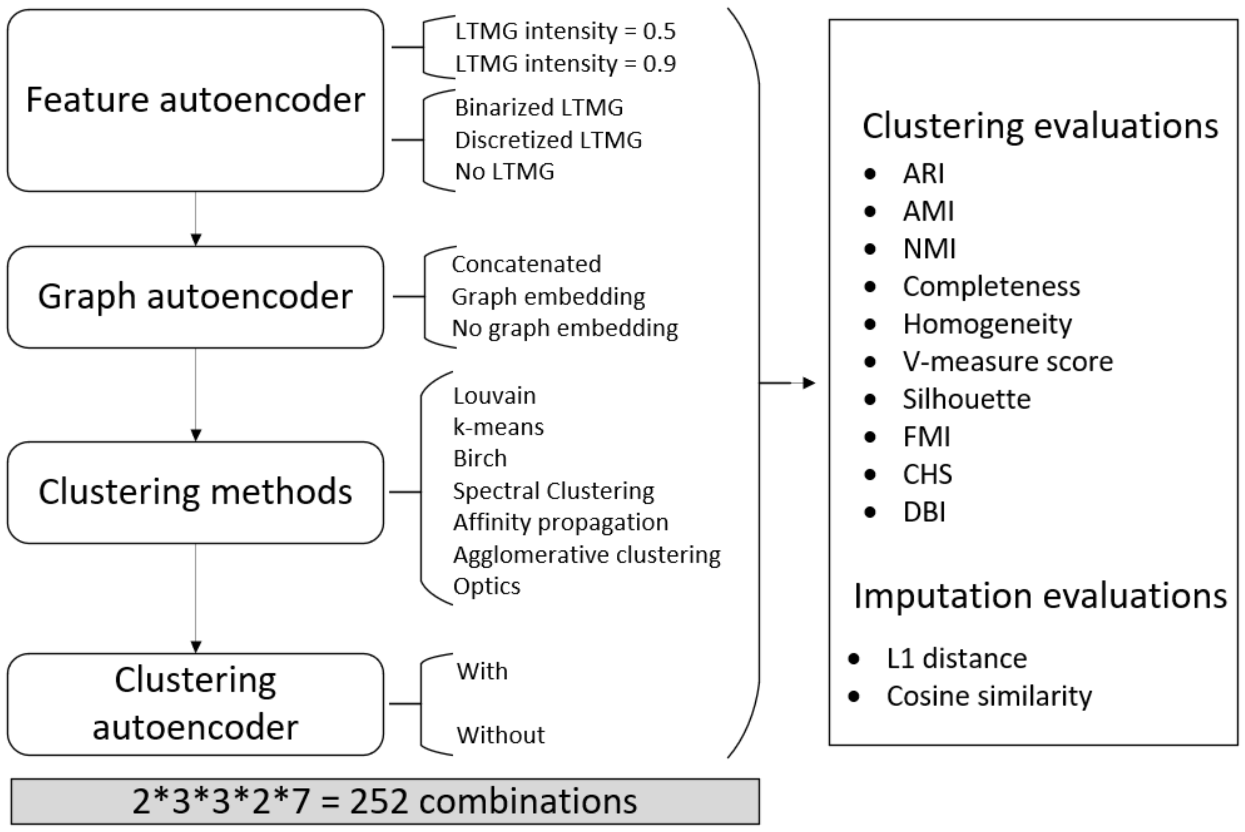


Figure S4. Design of ablation tests and parameter searching of scGNN.We tested 252 parameter combinations in scGNN for the three autoencoders in the iterative process to select the best scGNN performance. All results are evaluated via ten criteria for clustering and two for imputation.

4.11.2 Cell clustering results of scGNN

Besides the artificial dropout benchmarks, we continued to evaluate the clustering performance of scGNN and the seven imputation tools on the same two datasets.

Cell clustering and trajectory evaluations

(A) Comparison of ARI and Silhouette scores among scGNN and seven tools using Klein and Zeisel datasets (Figure 4A).

(B) Comparison of UMAP visualizations on the same two datasets, indicating that when scGNN embeddings are utilized, cells are more closely grouped within the same cluster but when other tools are used, cells are more separated between clusters. Raw data is clustered and visualized using Seurat (Figure 4B).

(C) Pseudotime analysis using the raw expression matrix and scGNN imputed matrix of the Klein dataset via Monocle2 (Figure 4C).

(D) Justification of using the graph autoencoder, the cluster autoencoder, and the top 2,000 variable genes on the Klein dataset in the scGNN framework, in terms of ARI. scGNN CA-shows the results of the graph autoencoder's ablation, CA-shows the results of the cluster autoencoder's ablation, and AG shows the results after using all genes in the framework (Figure 4D).

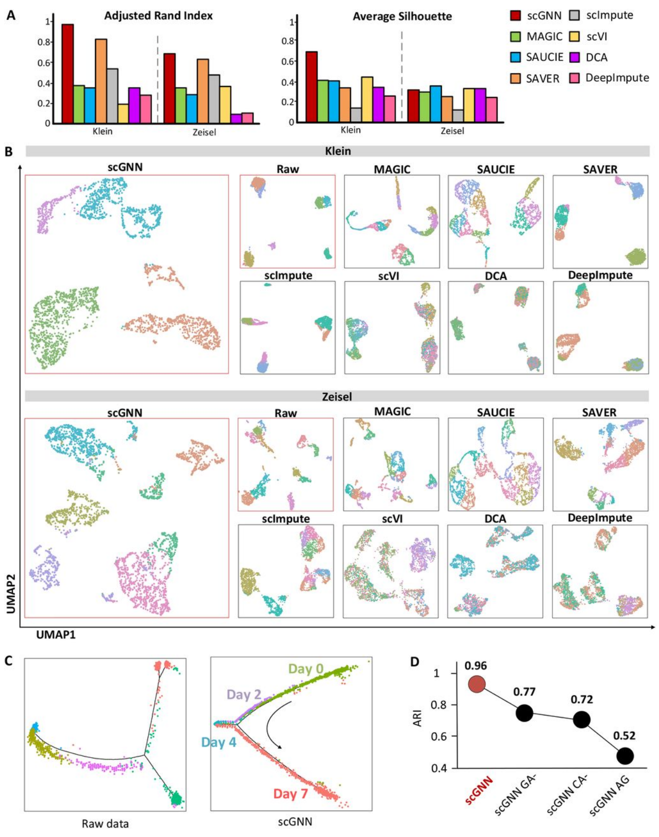


Figure 4. Cell clustering and trajectory evaluations.

Cell clustering results of scGNN compared to existing clustering tools

scGNN showed significant enhancement in cell clustering compared to the clustering tool (e.g., Seurat) when using the raw data. The comparison was conducted on four tools (i.e., Seurat, CIDR, RaceID, and Monocle3) using four benchmark datasets. ARI of each test is indicated on each UMAP, comparing the predicted cell clusters to the benchmark labels.

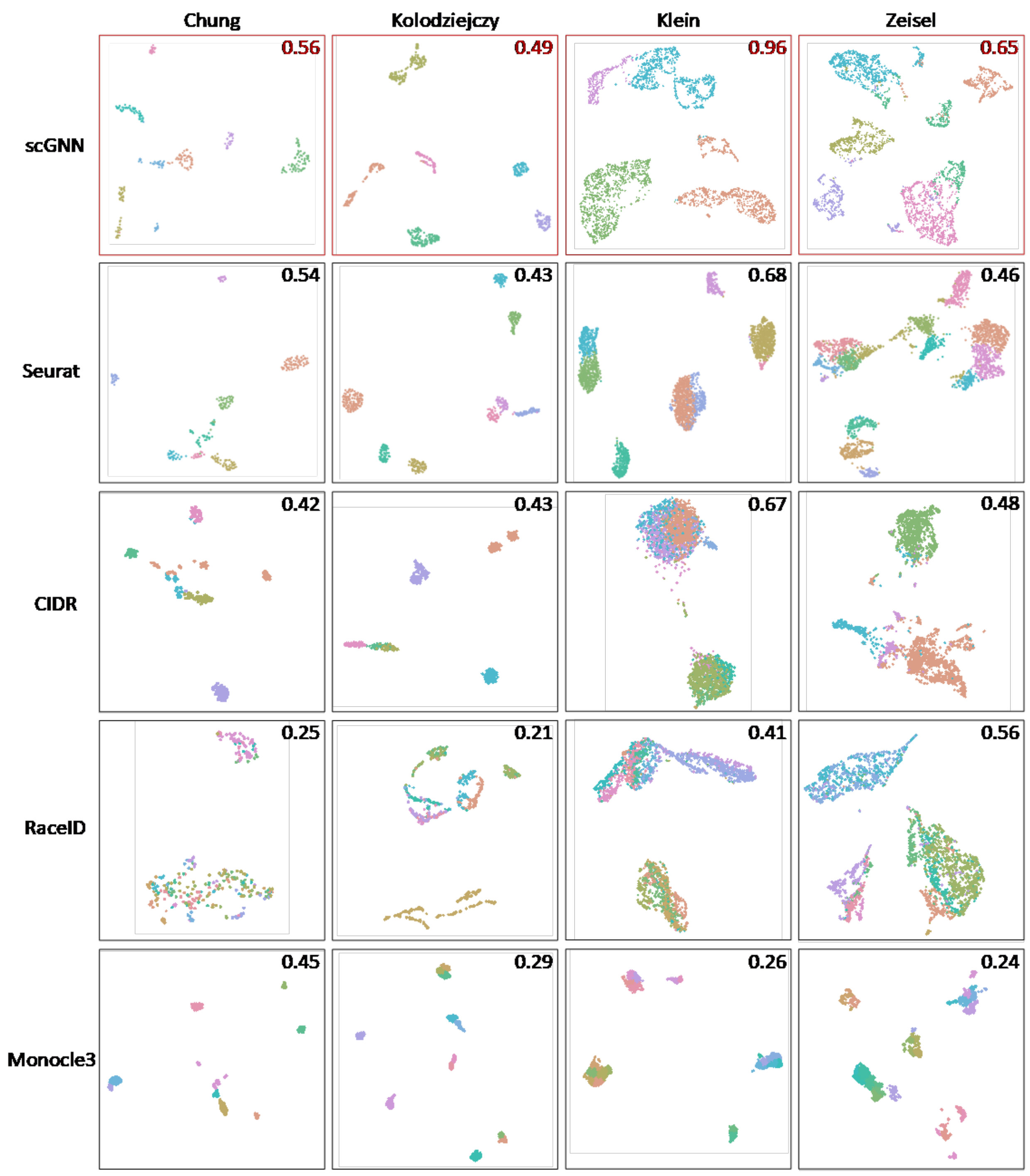


Figure S5. Clustering results of scGNN compared to existing clustering tools.

4.11.3 scGNN illustrates AD-related neural development and the underlying regulatory mechanism

To further demonstrate the applicative power of scGNN, we applied it to a scRNA-Seq dataset (GEO accession number GSE138852) containing 13,214 single nuclei collected from six AD and six control brains. scGNN identifies 10 cell clusters, including microglia, neurons, oligodendrocyte progenitor cells (OPCs), astrocytes, and six sub-clusters of oligodendrocytes (Figure 5A). Specifically, the proportions of these six oligodendrocyte sub-clusters differ between AD patients (Oligos 2, 3, and 4) and healthy controls (Oligos 1, 5, and 6) (Figure 5B). Moreover, the difference between AD and the control in the proportion of astrocyte and OPCs is observed, indicating the change of cell population in AD patients compared to healthy controls (Figure 5B). We then combined these six oligodendrocyte sub-clusters into one to discover DEGs. Since scGNN can significantly increase true signals in the raw dataset, DEG patterns are more explicit (Figure S6). Among all DEGs, we confirmed 22 genes as cell-type-specific markers for astrocytes, OPCs, oligodendrocytes, and neurons, in that order35 (Figure 5C). A biological pathway enrichment analysis shows several highly positive-enrichments in AD cells compared to control cells among all five cell types. These enrichments include oxidative phosphorylation and pathways associated with AD, Parkinson's disease, and Huntington disease36 (Figure 5D). Interestingly, we observed a strong negative enrichment of the MAPK (mitogen-activated protein kinase) signaling pathway in the microglia cells, suggesting a relatively low MAPK regulation in microglia than other cells.

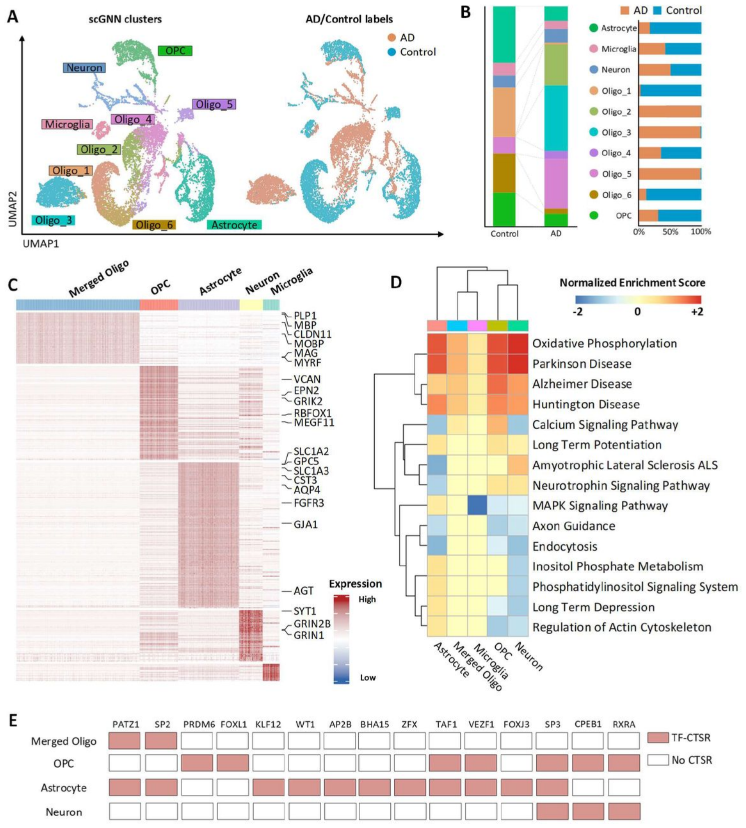


Figure 5 Alzheimer's disease dataset (GSE138852) analysis based on scGNN. (A) Cell clustering UMAP. Labeled with scGNN clusters (left) and AD/control samples (right). (B) Comparison of cell proportions in AD/control samples (left) and each cluster (right). (C) Heatmap of DEGs (logFC > 0.25) in each cluster. Six oligodendrocyte sub-clusters are merged as one to compare with other cell types. Marker genes identified in DEGs are listed on the right. (D) Selected AD-related enrichment pathways in each cell type in the comparison between AD and control cells. (E) Underlying TFs are responsible for the cell-type-specific gene regulations identified by IRIS3.

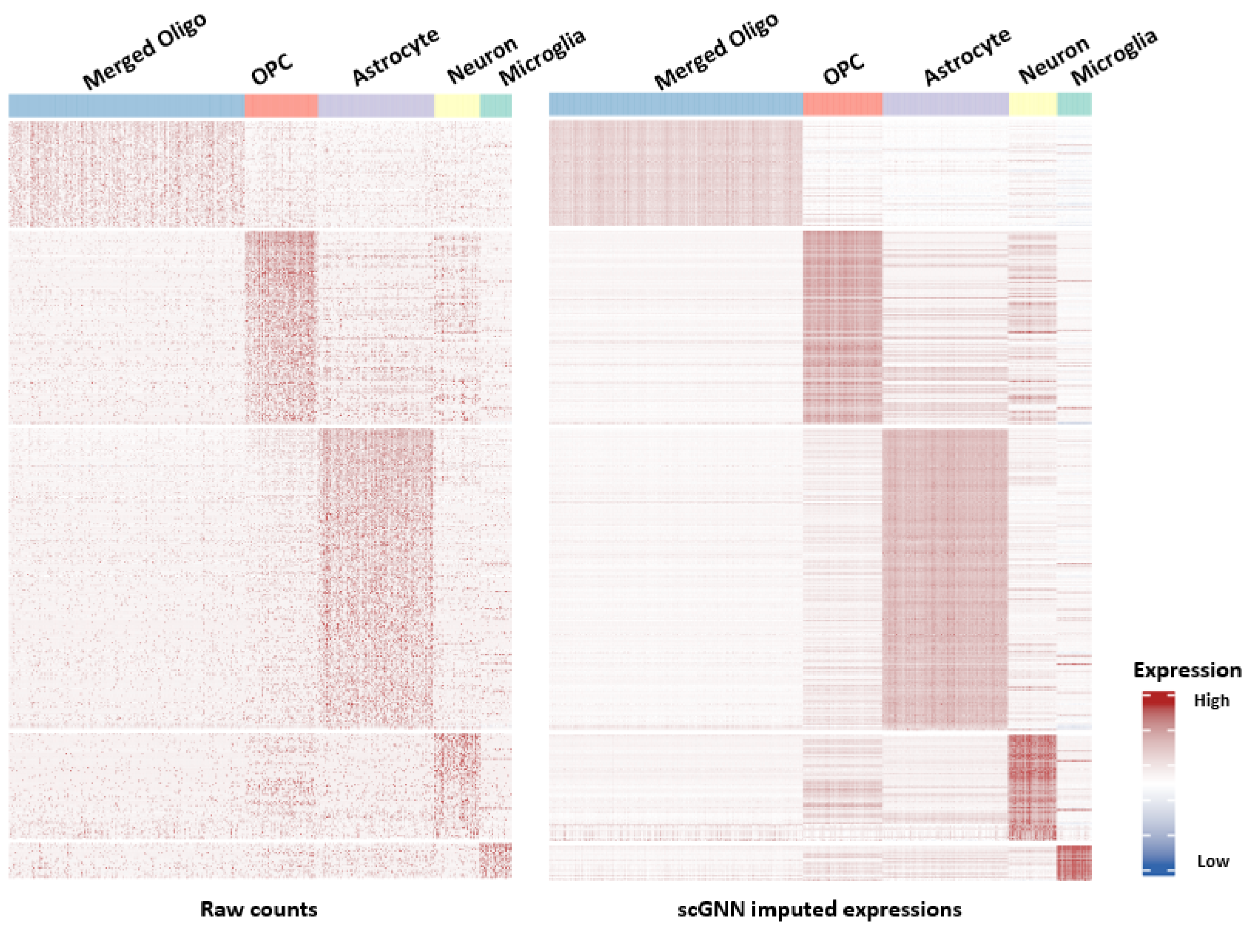


Figure S6. Comparison of DEG expression before (Left) and after scGNN imputation (Right). DEGs were identified using the Seurat package based on scGNN predicted clusters, and the six oligodendrocyte sub-clusters were merged into one. Cells were randomly selected from half of the merged oligo group to make the figure more balanced.

4.12 Benchmark data

The banchmark Chung data, Klein data, Zeisel data can be downloaded from GEO databases with accession numbers of:

- GSE75688 (the Chung data);
- GSE65525 (the Klein data);
- GSE60361 (the Zeisel data).

The banchmark Kolodziejczy data can be accessed from EBI with an accession number of E-MTAB-2600.

4.13 Casestudy

AD case datasets

To further demonstrate the applicative power of scGNN, we applied it to a scRNA-Seq dataset (GEO accession number GSE138852) containing 13,214 single nuclei collected from six AD and six control brains.

AD case datasets can be downloaded from GEO databases with accession numbers of GSE138852 (AD case).