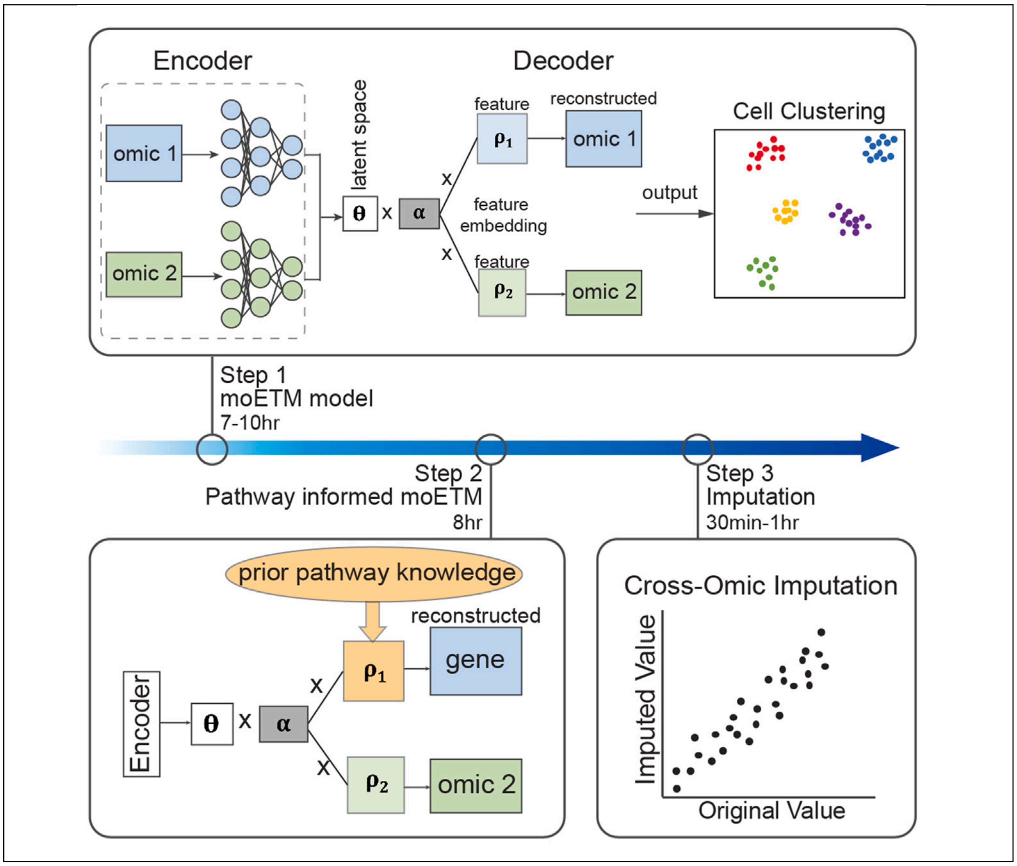
STAR Protocols

CellPress OPEN ACCESS

Protocol

Protocol to perform integrative analysis of high-dimensional single-cell multimodal data using an interpretable deep learning technique



Manqi Zhou, Hao Zhang, Zilong Bai, Dylan Mann-Krzisnik, Fei Wang, Yue Li

yueli@cs.mcgill.ca

Highlights

Steps for integrating multiple modalities to a low-dimensional representation

Instructions to accurately achieve cross-omics imputation

Guidance on incorporating prior pathway knowledge to improve interpretability

The advent of single-cell multi-omics sequencing technology makes it possible for researchers to leverage multiple modalities for individual cells. Here, we present a protocol to perform integrative analysis of high-dimensional single-cell multimodal data using an interpretable deep learning technique called moETM. We describe steps for data preprocessing, multi-omics integration, inclusion of prior pathway knowledge, and cross-omics imputation. As a demonstration, we used the single-cell multi-omics data collected from bone marrow mononuclear cells (GSE194122) as in our original study.

Publisher's note: Undertaking any experimental protocol requires adherence to local institutional guidelines for laboratory safety and ethics.

Zhou et al., STAR Protocols 5, 103066
June 21, 2024 © 2024 The Author(s). Published by Elsevier Inc.
https://doi.org/10.1016/j.xpro.2024.103066

STAR Protocols

CellPress
OPEN ACCESS

Protocol

Protocol to perform integrative analysis of high-dimensional single-cell multimodal data using an interpretable deep learning technique

Manqi Zhou,[1,2,7] Hao Zhang,[3,7] Zilong Bai,[2,3] Dylan Mann-Krzisnik,[4] Fei Wang,[2,3] and Yue Li[4,5,6,8,9,*]

[1]Department of Computational Biology, Cornell University, Ithaca, NY 14853, USA

[2]Institute of Artificial Intelligence for Digital Health, Weill Cornell Medicine, New York, NY 10021, USA

[3]Division of Health Informatics, Department of Population Health Sciences, Weill Cornell Medicine, New York, NY 10021, USA

[4]Quantitative Life Science, McGill University, Montréal, QC H3A 0G4, Canada

[5]School of Computer Science, McGill University, Montréal, QC H3A 0G4, Canada

[6]Mila – Quebec AI Institute, Montréal, QC H2S 3H1, Canada

[7]These authors contributed equally

[8]Technical contact

[9]Lead contact

*Correspondence: yueli@cs.mcgill.ca

https://doi.org/10.1016/j.xpro.2024.103066

SUMMARY

The advent of single-cell multi-omics sequencing technology makes it possible for researchers to leverage multiple modalities for individual cells. Here, we present a protocol to perform integrative analysis of high-dimensional single-cell multimodal data using an interpretable deep learning technique called moETM. We describe steps for data preprocessing, multi-omics integration, inclusion of prior pathway knowledge, and cross-omics imputation. As a demonstration, we used the single-cell multi-omics data collected from bone marrow mononuclear cells (GSE194122) as in our original study.

For complete details on the use and execution of this protocol, please refer to Zhou et al.[1]

BEFORE YOU BEGIN

This section includes the software installation as well as the data collection.

Hardware requirement

The implementation of moETM requires GPU usage. In this protocol, we used a GPU (Tesla P100-PCIE-16GB), a CPU with 32 cores and 257 GB RAM on Linux (Rocky 9.0), and the cuda version is cuda_11.8.r11.8.

Software installation

Timing: <30 min

1. Clone the code repository https://github.com/manqizhou/moETM as following:

```
> git clone https://github.com/manqizhou/moETM.git.
```

STAR Protocols 5, 103066, June 21, 2024 © 2024 The Author(s). Published by Elsevier Inc.

2. Create folders to store results:

```
> mkdir data

> mkdir result_fig

> mkdir Result

> mkdir Trained_model
```

3. Install packages according to the requirements file as following:

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

4. Install PyTorch from the website https://pytorch.org/ by choosing the one that matches your CUDA version. The original study was conducted under CUDA 11.8.

```
> conda install pytorch torchvision torchaudio pytorch-cuda=11.8 -c pytorch -c nvidia
```

Data collection

⏱ Timing: <30 min

Single-cell RNA sequencing (scRNA-seq) combined with assay for transposase-accessible chromatin using sequencing (ATAC-seq) simultaneously measure the transcriptome and chromatin accessibility in the same cells.[2] Cellular indexing of transcriptomes and epitopes by sequencing (CITE-seq) measures surface protein and transcriptome data using oligonucleotide-labeled antibodies.[3] Here, we take either scRNA-seq + scATAC-seq (gene+peak case) or CITE-seq (gene+protein case) in the h5ad format as the input of moETM.

5. Download scRNA-seq + scATAC-seq data (GSE194122_openproblems_neurips2021_multiome_BMMC_processed.h5ad.gz) and CITE-seq (GSE194122_openproblems_neurips2021_cite_BMMC_processed.h5ad.gz) from the GSE194122 website https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE194122.
6. Preprocess data to h5ad format.

   Note: The GSE194122 data is already in h5ad format, so no further steps are needed. If the downloaded data is not in the h5ad format, e.g. the mouse kidney cell sci-CAR data (GSE117089)[4] used in the original study, it needs to be transformed to h5ad format. The h5ad data format comprises three key components: the expression value matrix X, the cell information stored in the obs attribute, and the feature information stored in the var attribute. For data not stored in the h5ad format, we need to create an h5ad format and then manually assign the X and obs/var attributes using the python pakcage AnnData. The expression value matrix X and feature information are typically contained within the gene/peak/protein count file. The cell information is stored either in the barcode or cell type file. The detailed implementation of converting a non-h5ad format GSE117089 data to an h5ad format data for is in the script ./useful_file/mouse_brain_preprocess.py.

   a. Download data from https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE140203.

b. Read count files for gene expression and chromatin accessibility separately and then re-format them to h5ad format

```
> adata = ad.AnnData(X=X,obs=pd.DataFrame(index=GEX_barcode_new), var=pd.DataFrame
(index=var_name))

> adata = ad.AnnData(X=X, obs=pd.DataFrame(index=ATAC_barcode_new),var=pd.DataFrame
(index=var_name))
```

7. Put the preprocessed data into the data folder.
8. (Optional) To make the downstream analysis more interpretable, the user can filter genes by selecting protein-coding genes only. The example file for the BMMC dataset is under ./useful_files/gene_coding_nips_rna_protein.csv and gene_coding_nips_rna_atac.csv.

KEY RESOURCES TABLE

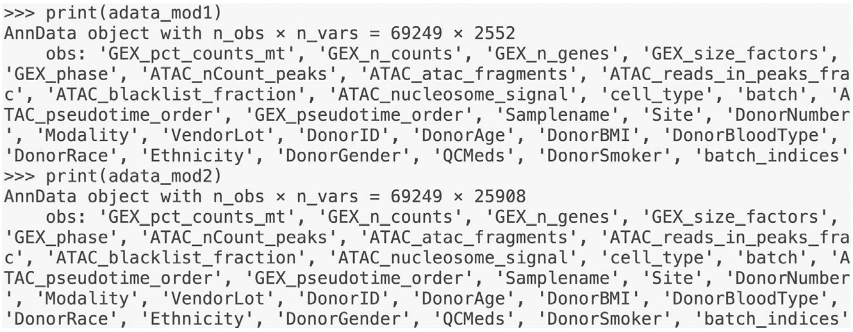
REAGENT or RESOURCE	SOURCE	IDENTIFIER
Deposited data		
BMMC CITE-seq	Luecken et al.[5]	GSE194122
BMMC multiome data	Luecken et al.[5]	GSE194122
Mouse skin cell SHARE-seq	Ma et al.[6]	GSE140203
Mouse brain cell SHARE-seq	Ma et al.[6]	GSE140203
Mouse kidney cell sci-CAR	Cao et al.[5]	GSE117089
PBMC CITE-seq	Hao et al.[7]	GSE164378
COVID-19 PBMC CITE-seq	Stephenson et al.[8]	https://www.covid19cellatlas.org/
Software and algorithms		
moETM	This paper	https://github.com/manqizhou/moETM
Scanpy 1.9.1	Wolf et al.[9]	https://github.com/scverse/scanpy
AnnData 0.8.0	Virshup et al.[10]	https://github.com/scverse/anndata
biomaRt 2.46.3	Durinck et al.[10]	https://rdrr.io/bioc/biomaRt/man/
Seurat 4.3.0	Hao et al.[7]	https://satijalab.org/seurat/
totalVI	Gayoso et al.[7,11]	https://github.com/YosefLab/scvi-tools
SMILE	Xu et al.[12]	https://github.com/rpmccordlab/SMILE
scMM	Minoura et al.[13]	https://github.com/kodaim1115/scMM
Cobolt	Gong et al.[13,14]	https://github.com/epurdom/cobolt
MultiVI	Ashuach et al.[15]	https://zenodo.org/record/5762077
MOFA+	Argelaguet et al.[16]	https://github.com/bioFAM/MOFA2
Python 3.9.5	Python Software Foundation[17]	http://www.python.org/
R 4.0.5	R Core Team[18]	https://www.r-project.org/
Other		
GPU	N/A	Tesla P100-PCIE-16GB

STEP-BY-STEP METHOD DETAILS

Here, we describe step-by-step methods for analyzing single cell multi-omics data, i.e., the gene+peak data and the gene+protein data, including the: 1) data preprocessing, 2) integration task, 3) inclusion of prior pathway knowledge, and 4) imputation task. To illustrate these various steps, we use the BMMC1 data (gene+peak) and the BMMC2 data (gene+protein) from GSE194122 as an example. All steps can be found in the repository https://github.com/manqizhou/moETM/tree/main.

Data preprocessing

⏱ Timing: <30 min



```
>>> print(adata_mod1)
AnnData object with n_obs × n_vars = 69249 × 2552
    obs: 'GEX_pct_counts_mt', 'GEX_n_counts', 'GEX_n_genes', 'GEX_size_factors',
'GEX_phase', 'ATAC_nCount_peaks', 'ATAC_atac_fragments', 'ATAC_reads_in_peaks_fra
c', 'ATAC_blacklist_fraction', 'ATAC_nucleosome_signal', 'cell_type', 'batch', 'A
TAC_pseudotime_order', 'GEX_pseudotime_order', 'Samplename', 'Site', 'DonorNumber
', 'Modality', 'VendorLot', 'DonorID', 'DonorAge', 'DonorBMI', 'DonorBloodType',
'DonorRace', 'Ethnicity', 'DonorGender', 'QCMeds', 'DonorSmoker', 'batch_indices'
>>> print(adata_mod2)
AnnData object with n_obs × n_vars = 69249 × 25908
    obs: 'GEX_pct_counts_mt', 'GEX_n_counts', 'GEX_n_genes', 'GEX_size_factors',
'GEX_phase', 'ATAC_nCount_peaks', 'ATAC_atac_fragments', 'ATAC_reads_in_peaks_fra
c', 'ATAC_blacklist_fraction', 'ATAC_nucleosome_signal', 'cell_type', 'batch', 'A
TAC_pseudotime_order', 'GEX_pseudotime_order', 'Samplename', 'Site', 'DonorNumber
', 'Modality', 'VendorLot', 'DonorID', 'DonorAge', 'DonorBMI', 'DonorBloodType',
'DonorRace', 'Ethnicity', 'DonorGender', 'QCMeds', 'DonorSmoker', 'batch_indices'
```

Figure 1. Screenshot of the preprocessed BMMC1 data structure

'adata_mod1' and 'adata_mod2' represent the preprocessed gene and the peak modality data, respectively. Following the preprocessing procedure in 9a, only the expression matrix and the 'obs' attribute were retained.

In this section, we describe steps for data preprocessing. moETM requires cell-by-feature matrices as input, where features could be gene, protein, or peak. The input data is in the AnnData format and is loaded and preprocessed by the load_*_dataset() and prepare_*_dataset() functions in the dataloader.py script. Before putting into the model, all matrices are column normalized by dividing the column sum.

1. For the BMMC1 (GSE194122_openproblems_neurips2021_multiome_BMMC_processed.h5ad) and BMMC2 (GSE194122_openproblems_neurips2021_cite_BMMC_processed.h5ad) data which are downloaded in the step 5 and used in the original study, they are preprocessed by functions in the dataloader.py.
   a. For the BMMC1 (gene+peak case) data, both modalities underwent a 3-step preprocessing procedure: normalize counts per cell by scanpy.pp.normalize_total, logarithmize the data matrix by scanpy.pp.log1p, and select highly variable features by scanpy.pp.highly_variable_genes. The 3 steps were incorporated within the load_nips_rna_atac_dataset() function.

```
> adata_mod1, adata_mod2 = load_nips_rna_atac_dataset(mod_file_path, gene_encoding)

> adata_mod1, adata_mod2 = prepare_nips_dataset(adata_mod1, adata_mod2)
```

Note: Before input the preprocessed data into the model, we retained only the obs attribute, which contains cell type information and batch information, and removed other attributes. This was accomplished by generating a new h5ad object, assigning the preprocessed data matrix as X, and incorporating cell type and batch information into the obs attribute. This step was incorporated within the prepare_nips_dataset() function.

   The preprocessed data structure was shown in the Figure 1.
   b. For the BMMC2 (gene+peak case) data, the gene data underwent the same 3 preprocessing steps as described in Step 1a.

Note: The protein data was subjected to the first two steps. All proteins data was utilized without selecting highly variable proteins. This was motivated by the substantially smaller number of proteins (~100) compared to the number of genes (~14000) and peaks (~110000). Similar to the BMMC1 case, the preprocessed steps were all involved within the load_nips_rna_protein_dataset() function.

```
> adata_mod1, adata_mod2 = load_nips_dataset_rna_protein_dataset(mod_file_path, gene_encoding, protein_encoding)

> adata_mod1, adata_mod2 = prepare_nips_dataset(adata_mod1, adata_mod2)
```

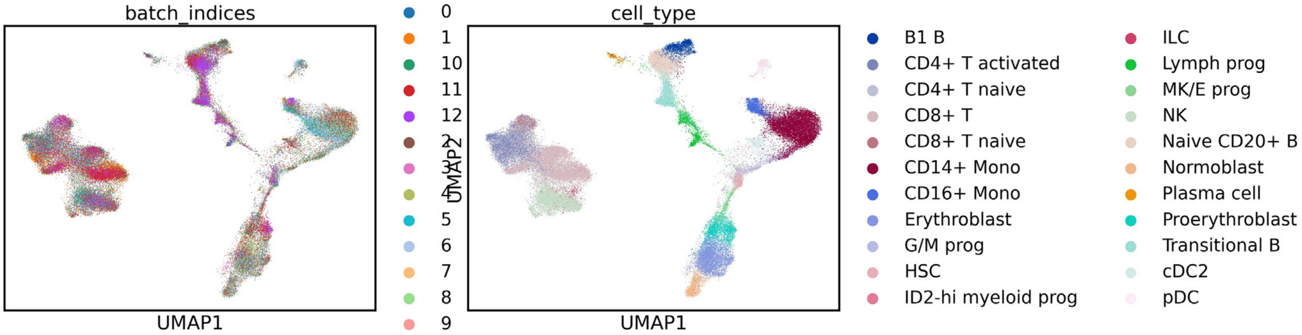


Figure 2. UMAP visualization
Each dot represents a cell. The left panel is colored by batch indexes and the right panel is colored by cell types.

The output of the step 1 is 2 preprocessed h5ad format data containing cell type information and batch information. This step is already included in the beginning of scripts for steps 2 and 3.

Multi-omics integration

Timing: 7–10 h

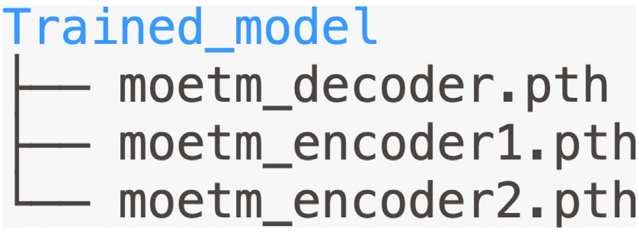
In this section, we describe steps for the implementation of multi-omics integration.

2. Train moETM to integrate single cell gene expression and surface protein data. Here, we used the BMMC2 data as the original study.
   a. The input data will first be preprocessed following step 9.
   b. Prepare training and evaluation data by the function data_process_moETM().

```
> X_mod1_train_T, X_mod2_train_T, batch_index_train_T, train_adata_mod1 = data_process_
moETM(adata_mod1, adata_mod2)
```

   c. Set hyper parameters num_topic, emd_dim, batch_size, Total_epoch.

Note: num_topic is the number of topics that the model will learn. It depends on the complexity of the dataset. emd_dim refers to the embedding dimension, which is the size of the vector representations for each topic. A higher dimension can capture more information but may also lead to overfitting and increased computational cost. batch_size is the number of training examples used in one iteration of the training process. Smaller batch sizes can lead to faster convergence but may be noisier, while larger batch sizes provide more stable gradients but require more memory. total_epoch is the number of times the entire dataset is passed



```
Trained_model
├── moetm_decoder.pth
├── moetm_encoder1.pth
└── moetm_encoder2.pth
```

Figure 3. Folder structure of the saved trained model
The trained model was saved in the Trained_model folder containing three files: moetm_encoder1/2.pth for the omic-specific encoders parameters and moetm_decoder.pth for the decoder parameters.

Table 1. Example of the outcome moetm_all_data.csv file

Epoch	ARI	NMI	ASW	ASW_2	B_kBET	B_ASW	B_GC	B_ebm
0	0.49	0.65	0.06	0.53	0.05	0.88	0.95	1.07
10	0.53	0.67	0.07	0.53	0.04	0.89	0.96	0.95
20	0.47	0.67	0.07	0.54	0.03	0.92	0.96	0.86
30	0.48	0.67	0.07	0.54	0.01	0.93	0.96	0.77
40	0.50	0.68	0.06	0.53	0.01	0.93	0.95	0.74
50	0.57	0.72	0.07	0.54	0.01	0.94	0.96	0.69

The first 4 columns represent metrics for assessing bio-conservation, while the last four columns, denoted by 'B_', refer to metrics evaluating the effects of batch-effect removal. The evaluation metrics are adopted from a previous study.[19]

through the model during training. More epochs can lead to better training, but also increase the risk of overfitting. In our study, we set num_topic = 100, emd_dim = 400, batch_size = 2000, Total_epoch = 500. The optimal num_topic was selected based on the robust performance, where the performance of the model (e.g., ARI value) did not increase much after num_topic > 100. The batch_size and total_epoch was chosen based the computational capacity considerations. The model's performance stabilizes after 500 epochs. For large dataset, the user might increase the epoch number.

d. Train the model using Train_moETM().

```
> Train_moETM(trainer, Total_epoch, train_num, batch_size, Train_set, Test_set, Eval_kwargs)
```

The implementation is included in main_integration_rna_protein.py.

3. Train moETM to integrate single cell gene expression and chromatin accessibility data. Here, we used the BMMC1 data as the original study.
   a. The input data will first be preprocessed following step 9.
   b. Prepare training and evaluation data by the function data_process_moETM().

```
> X_mod1_train_T, X_mod2_train_T, batch_index_train_T, train_adata_mod1 = data_process_
moETM(adata_mod1, adata_mod2)
```

   c. Set hyper parameters num_batch, num_topic, emd_dim, batch_size, Total_epoch.

Note: The num_topic, emd_dim, batch_size, Total_epoch is the same as described in the step 2c. num_batch is the number of batches during the training process. It is directly related with batch_size, and equals to ceil(#total samples/batch_size), where the ceil function returns the smallest integer value which is greater than or equal to the specified number. The users may adjust batch_size based on their model performance and hardware limitations.

   d. Train the model using Train_moETM().

```
> Train_moETM(trainer, Total_epoch, train_num, batch_size, Train_set, Test_set, Eval_kwargs)
```

The implementation is included in main_integration_rna_atac.py. The difference between step 2 and step 3 is the way of preprocessing different data.

The step 2/3 will produce 3 outputs: 1) evaluation metrics values saved in ./Results/moetm_all_data.csv (Table 1), 2) UMAP visualization of the integrated low-dimensional representation stored in ./result_fig (Figure 2), 3) the trained model saved in ./Trained_model (Figure 3). Here we present the results of the BMMC1 data from step 3.

The moetm_all_data.csv file has 9 columns and 50 rows. The first column is the epoch number, while the remaining 8 columns contain evaluation matrices related to bio-conservation and batch-effect removal. Given our total of 500 epochs and the storage of results every 10 epochs, the file contains 50 rows. Table 1 is the first 6 rows of the file.

Except for evaluation metrics, we also plotted a UMAP visualization of the model per 10 epochs. Therefore, in the results_fig folder, we have 50 figures. Figure 2 is an example in epoch 490.

The trained model was stored under the folder Trained_model as shown in Figure 3.

4. Additionally, all scripts can be submitted as a batch job using a job scheduler such as "slurm" to a GPU node. To achieve this, the user needs to create a shell script containing commands involved in executing the job. For example, the user can create a script named submit.sh containing the following lines (comments inside the parentheses should be deleted),

```
> #!/bin/bash -l
> #SBATCH --mem=32000 (set the memory; here 32GB)
> #SBATCH --time=00:09:00 (set the time; here 9hr)
> #SBATCH --cluster cbsugpu03 (set the used cluster)
> #SBATCH --gres=gpu:tP100:1 (set the used GPU)
> #SBATCH --chdir=/workdir/mz335/moETM/github/ (set the work path)
> #SBATCH --job-name=moETM (set name of job)
> #SBATCH --output=moETM.out (write stdout+stderr)
> #SBATCH --mail-user=mz335@cornell.edu
> #SBATCH --mail-type=ALL (sent email at job start/end/crash)
> conda activate covid (activate conda encironment)
> module load python
> python main_intergration_rna_protein.py (run the script)
```

Then execute the sbatch command to submit the job,

```
> sbatch submit.sh
```

Inclusion of prior pathway knowledge

Timing: 8 h

In this section, we describe steps for how to include prior pathway knowledge in the model.

moETM can use prior pathway knowledge information by adding a pathway-by-gene matrix in the encoder. We downloaded pathways from MSigDB and selected the C7: immunologic signature gene sets as the BMMC data is related to immune cells.[20] We kept pathways that contain more than 5 and fewer than 1000 genes as pathways containing too few or too many genes might introduce noise or impose a computation burden based on the previous study.[21]

5. Download MSigDB gene sets from https://www.gsea-msigdb.org/gsea/msigdb/human/collections.jsp.

6. Filter pathways that have more than 5 and less than 1000 genes. Then prepare a binary pathway by gene matrix while 1 indicating the gene is in the pathway.

   Note: The generated binary matrix will serve as the feature embedding matrix in the topic modeling. In the default model used previously, the feature embedding matrix consist of learnable parameters. The inclusion of prior pathway knowledge allows us to directly map each topic to each gene set, which may further improve the model interpretability. As the BMMC data were derived from the bone marrow, we used the Immunologic signature gene sets collection (C7) here. The user might select the gene sets collection that aligns with their specific data.

```
> python generate_gene_pathway_biMatrix.py
```

Save and move the output csv file to the ./useful_file folder.

7. Train the moETM using prior pathway knowledge.

```
> python main_integration_rna_atac_use_pathway.py
```

Cross-omics imputation

Timing: 30 min - 1 h

In this section, we describe steps of cross-omics imputation.

8. Create recon folder to store results.

```
> mkdir recon
```

9. Impute gene expression from surface protein values.

   Note: The train_num is the number of training sample. Here, the default hyperparameters were set as the same as the step 2c. We set the Total_epoch = 500, batch_size = 2000 based on our computation capability.

```
> direction = ’another_to_rna’

> Train_moETM_for_cross_prediction(trainer, Total_epoch, train_num, batch_size, Train_set, Test_set)
```

To impute protein values from gene expression, change direction = ’rna_to_another’. The implementation is included in main_cross_prediction_rna_protein.py.

10. Impute gene expression value from peak values.

```
> direction = ’another_to_rna’

> Train_moETM_for_cross_prediction(trainer, Total_epoch, train_num, batch_size, Train_set, Test_set)
```

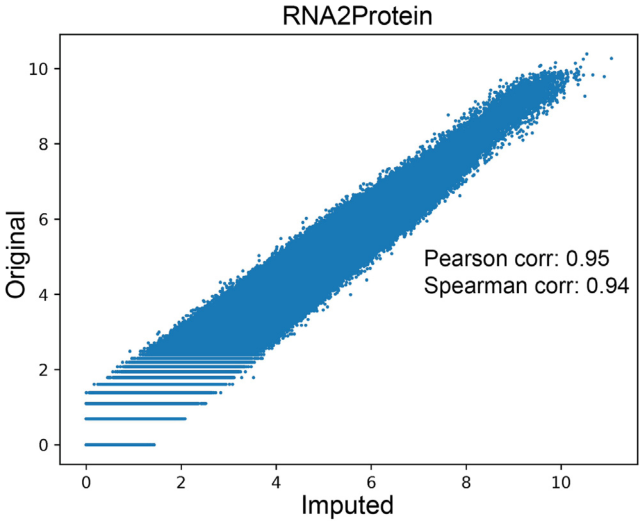


Figure 4. Scatterplot of original and imputed surface protein expression

The x-axis and y-axis represent the imputed and original protein expression values, respectively.

To impute chromatin accessibility values from gene expression, change direction = 'rna_to_another'. The implementation is included in main_cross_prediction_rna_atac.py.

The output was the imputed values. We then can plot a scatter plot to visualize the results.

EXPECTED OUTCOMES

In this protocol, we implemented a unified interpretable deep learning model called moETM to integrate single-cell multi-omics data including transcriptome and chromatin accessibility or surface proteins, which are the most common types of single-cell multi-omics data to date.

For the multi-omics integration results, each step of steps 2, 3, 7 will generate 3 outputs: 1) evaluation metrics values ./Results/moetm_all_data.csv (Table 1), 2) UMAP visualization of the integrated low-dimensional representation under ./result_fig (Figure 2); 3) saved trained model under ./Trained_model (Figure 3).

For the cross-omics imputation, each step of steps 9, 10 will generate a reconstructed value file under the recon folder (Figure 4).

LIMITATIONS

There are several challenges that are not addressed in moETM. For instance, moETM has the capacity to integrate across multiple batches and modalities as shown in step 3 and Table 1, but it requires the training data to have all omics measured in the same cells. A more challenging task is to integrate multimodal data without anchored features or cells, which is commonly known as the diagonal integration. Some recent approaches made use of graph representation learning to integrate multi-omics single-cell data at the expense of computational complexity and interpretability.

TROUBLESHOOTING

Problem 1

The Python package libraries are not supported in the existing environment when installing required dependents.

Potential solution

Create a virtual environment and follow specific versions of packages closely as instructed.

Problem 2

The modality data are not stored in the same file. moETM fails to read input data.

Potential solution

Preprocess the input data into h5ad format. Please refer to the corresponding Data preprocessing for details.

RESOURCE AVAILABILITY

Lead contact

Further information and requests for resources should be directed to and will be fulfilled by the lead contact, Yue Li (yueli@cs.mcgill.ca).

Technical contact

Questions about the technical specifics of performing the protocol should be directed to and will be answered by the technical contact, Yue Li (yueli@cs.mcgill.ca).

Materials availability

This study did not generate any reagents.

Data and code availability

- All data used in this study is publicly available. The peripheral blood mononuclear cells CITE-seq measuring from both COVID patients and healthy patients is available at the website https://www.covid19cellatlas.org/. The other datasets used are available under the NCBI GEO accession numbers as listed in the key resources table.
- All original code has been deposited at https://doi.org/10.5281/zenodo.8104798 and https://github.com/manqizhou/moETM and is publicly available as of the date of publication.
- Any additional information required to reanalyze the data reported in this paper is available from the lead contact upon request.

ACKNOWLEDGMENTS

F.W. would like to acknowledge the support from NIH R01AG076448, R01AG076234, RF1AG072449, RF1AG084178, NSF 1750326, and 2212175. Y.L. is supported by NSERC Alliance Catalyst ALLRP 576153-22, NSERC Discovery grant DGECR-2019-00253, and CIHR Canada Research Chair (Tier 2) in Machine Learning for Genomics and Healthcare.

AUTHOR CONTRIBUTIONS

M.Z. and H.Z. designed and performed the experiments under supervision of F.W. and Y.L. All authors read and approved the final manuscript.

DECLARATION OF INTERESTS

The authors declare no competing interests.

REFERENCES

1. Zhou, M., Zhang, H., Bai, Z., Mann-Krzisnik, D., Wang, F., and Li, Y. (2023). Single-cell multi-omics topic embedding reveals cell-type-specific and COVID-19 severity-related immune signatures. Cell Rep. Methods *3*, 100563. https://doi.org/10.1016/j.crmeth.2023.100563.
2. Buenrostro, J.D., Wu, B., Litzenburger, U.M., Ruff, D., Gonzales, M.L., Snyder, M.P., Chang, H.Y., and Greenleaf, W.J. (2015). Single-cell chromatin accessibility reveals principles of regulatory variation. Nature *523*, 486–490. https://doi.org/10.1038/nature14590.
3. Stoeckius, M., Hafemeister, C., Stephenson, W., Houck-Loomis, B., Chattopadhyay, P.K., Swerdlow, H., Satija, R., and Smibert, P. (2017). Simultaneous epitope and transcriptome measurement in single cells. Nat. Methods *14*, 865–868. https://doi.org/10.1038/nmeth.4380.

4. Cao, J., Cusanovich, D.A., Ramani, V., Aghamirzaie, D., Pliner, H.A., Hill, A.J., Daza, R.M., McFaline-Figueroa, J.L., Packer, J.S., Christiansen, L., et al. (2018). Joint profiling of chromatin accessibility and gene expression in thousands of single cells. Science *361*, 1380–1385. https://doi.org/10.1126/science.aau0730.

5. Luecken, M., Burkhardt, D., Cannoodt, R., Lance, C., Agrawal, A., Aliee, H., Chen, A., Deconinck, L., Detweiler, A., Granados, A., et al. (2021). A sandbox for prediction and integration of DNA, RNA, and proteins in single cells. Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks *1*.

6. Ma, S., Zhang, B., LaFave, L.M., Earl, A.S., Chiang, Z., Hu, Y., Ding, J., Brack, A., Kartha, V.K., Tay, T., et al. (2020). Chromatin Potential Identified by Shared Single-Cell Profiling of RNA and Chromatin. Cell *183*, 1103–1116.e20. https://doi.org/10.1016/j.cell.2020.09.056.

7. Hao, Y., Hao, S., Andersen-Nissen, E., Mauck, W.M., 3rd, Zheng, S., Butler, A., Lee, M.J., Wilk, A.J., Darby, C., Zager, M., et al. (2021). Integrated analysis of multimodal single-cell data. Cell *184*, 3573–3587.e29. https://doi.org/10.1016/j.cell.2021.04.048.

8. Stephenson, E., Reynolds, G., Botting, R.A., Calero-Nieto, F.J., Morgan, M.D., Tuong, Z.K., Bach, K., Sungnak, W., Worlock, K.B., Yoshida, M., et al. (2021). Single-cell multi-omics analysis of the immune response in COVID-19. Nat. Med. *27*, 904–916. https://doi.org/10.1038/s41591-021-01329-2.

9. Wolf, F.A., Angerer, P., and Theis, F.J. (2018). SCANPY: large-scale single-cell gene expression data analysis. Genome Biol. *19*, 15. https://doi.org/10.1186/s13059-017-1382-0.

10. Durinck, S., Moreau, Y., Kasprzyk, A., Davis, S., De Moor, B., Brazma, A., and Huber, W. (2005). BioMart and Bioconductor: a powerful link between biological databases and microarray data analysis. Bioinformatics *21*, 3439–3440. https://doi.org/10.1093/bioinformatics/bti525.

11. Gayoso, A., Steier, Z., Lopez, R., Regier, J., Nazor, K.L., Streets, A., and Yosef, N. (2021). Joint probabilistic modeling of single-cell multi-omic data with totalVI. Nat. Methods *18*, 272–282. https://doi.org/10.1038/s41592-020-01050-x.

12. Xu, Y., Das, P., and McCord, R.P. (2022). SMILE: mutual information learning for integration of single-cell omics data. Bioinformatics *38*, 476–486. https://doi.org/10.1093/bioinformatics/btab706.

13. Minoura, K., Abe, K., Nam, H., Nishikawa, H., and Shimamura, T. (2021). A mixture-of-experts deep generative model for integrated analysis of single-cell multiomics data. Cell Rep. Methods *1*, 100071. https://doi.org/10.1016/j.crmeth.2021.100071.

14. Gong, B., Zhou, Y., and Purdom, E. (2021). Cobolt: integrative analysis of multimodal single-cell sequencing data. Genome Biol. *22*, 351. https://doi.org/10.1186/s13059-021-02556-z.

15. Ashuach, T., Gabitto, M.I., Jordan, M.I., and Yosef, N. (2021). MultiVI: deep generative model for the integration of multi-modal data. bioRxiv. https://doi.org/10.1101/2021.08.20.457057.

16. Argelaguet, R., Arnol, D., Bredikhin, D., Deloro, Y., Velten, B., Marioni, J.C., and Stegle, O. (2020). MOFA+: a statistical framework for comprehensive integration of multi-modal single-cell data. Genome Biol. *21*, 111. https://doi.org/10.1186/s13059-020-02015-1.

17. Van Rossum, G., and Drake, F.L. (1995). Python reference manual (Centrum voor Wiskunde en Informatica Amsterdam).

18. R Core Team, R. (2013). R: A Language and Environment for Statistical Computing.

19. Luecken, M.D., Büttner, M., Chaichoompu, K., Danese, A., Interlandi, M., Müller, M.F., Strobl, D.C., Zappia, L., Dugas, M., Colomé-Tatché, M., and Theis, F.J. (2022). Benchmarking atlas-level data integration in single-cell genomics. Nat. Methods *19*, 41–50.

20. Godec, J., Tan, Y., Liberzon, A., Tamayo, P., Bhattacharya, S., Butte, A.J., Mesirov, J.P., and Haining, W.N. (2016). Compendium of Immune Signatures Identifies Conserved and Species-Specific Biology in Response to Inflammation. Immunity *44*, 194–206. https://doi.org/10.1016/j.immuni.2015.12.006.

21. Iorio, F., Garcia-Alonso, L., Brammeld, J.S., Martincorena, I., Wille, D.R., McDermott, U., and Saez-Rodriguez, J. (2018). Pathway-based dissection of the genomic heterogeneity of cancer hallmarks' acquisition with SLAPenrich. Sci. Rep. *8*, 6713.