CNS Summer Students 2024

General Project Results

Fan Huang

Million-level cell embedding Visualization

Start from visualization of two Lung cell datasets

hubmap id
HBM948.GXMD.986
HBM975.WQQQ.853

Cell number: 202k
Embedding space: 201905 × 60286

Special settings:
1. Utilize the correct matrix layer:

‘spliced_unspliced_sum’

2. Adopt two steps of normalization before conducting the UMAP function.

sc.pp.normalize_total(adata)
sc.pp.log1p(adata)

Same normalization settings for ALL 10 lung datasets

cell type number is: 48

CL_label	
capillary endothelial cell	119435
type I pneumocyte	94120
type II pneumocyte	51142
effector memory CD8-positive, alpha-beta T cell	42042
alveolar type 1 fibroblast cell	25102
multi-ciliated epithelial cell:non-nasal	24346
CD4-positive helper T cell	20184
ionocyte	14056
endothelial cell of venule	11042
endothelial cell of artery	10208
lung pericyte	6133
alveolar capillary type 2 endothelial cell	4925
alveolar type 2 fibroblast cell	4324
respiratory basal cell:resting	3358
smooth muscle cell	3066
endothelial cell of lymphatic vessel:mature	2203
endothelial cell of venule:pulmonary	2065
monocyte	1983
CD1c-positive myeloid dendritic cell	1877
B cell	1624

Million-level cell embedding Visualization

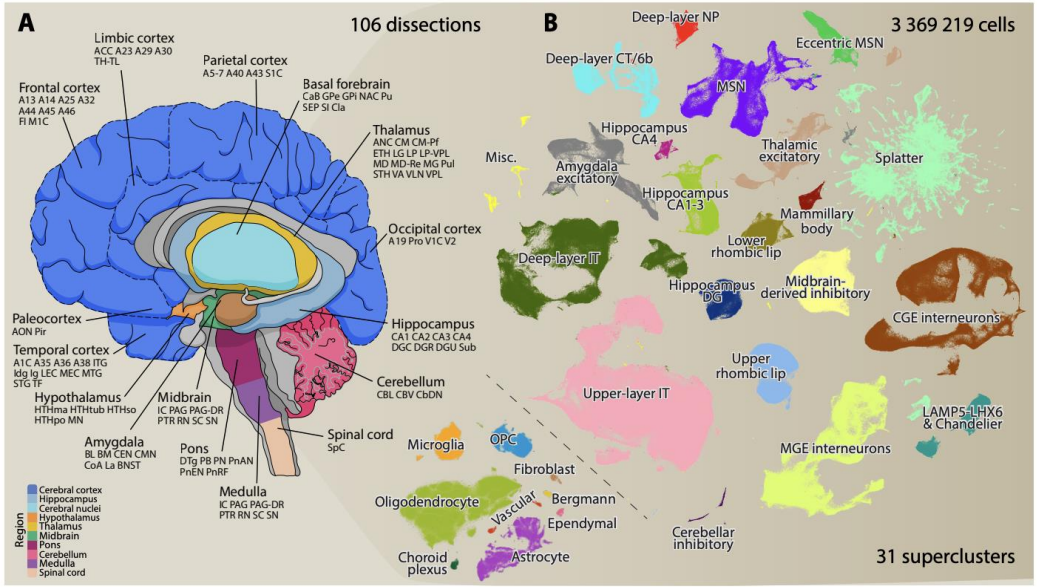
Alternative Visualization Published work based on PCA + t-SNE

Transcriptomic diversity of cell types across the adult human brain

Kimberly Siletti, Rebecca Hodge, Alejandro Mossi Albiach, Lijuan Hu, Ka Wai Lee, Peter Lönnerberg, Trygve Bakken, Song-Lin Ding, Michael Clark, Tamara Caspe
Julie Nyhus, Herman Tung, Anna Marie Yanny, Ernest Arenas, Ed

doi: https://doi.org/10.1101/2022.10.12.511898

Now published in *Science* doi: 10.1126/science.add7046

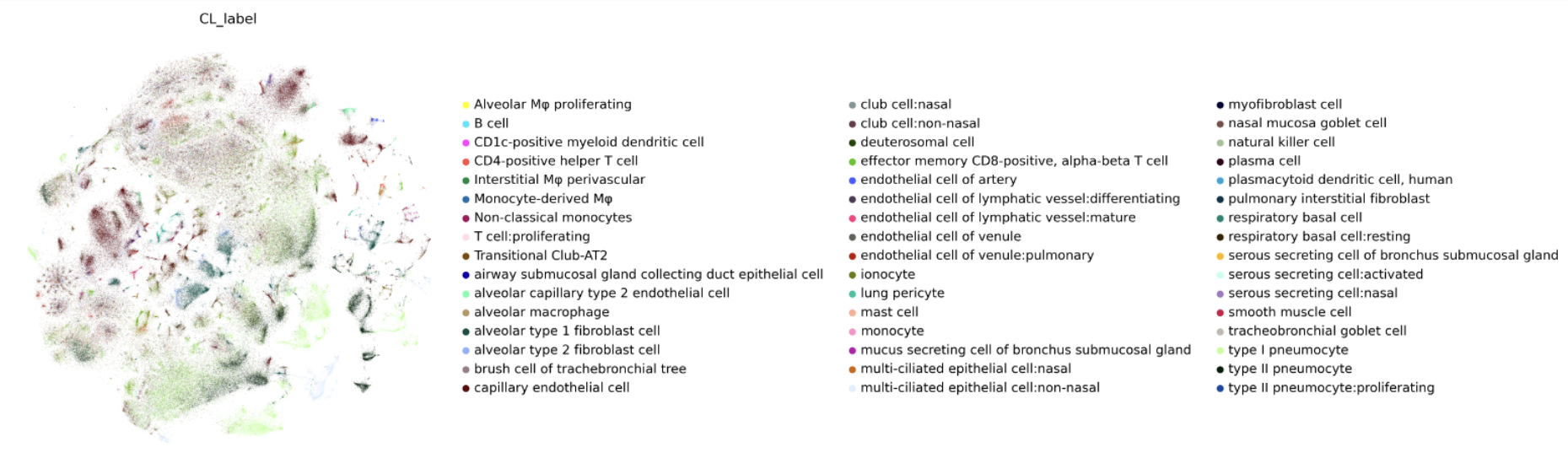


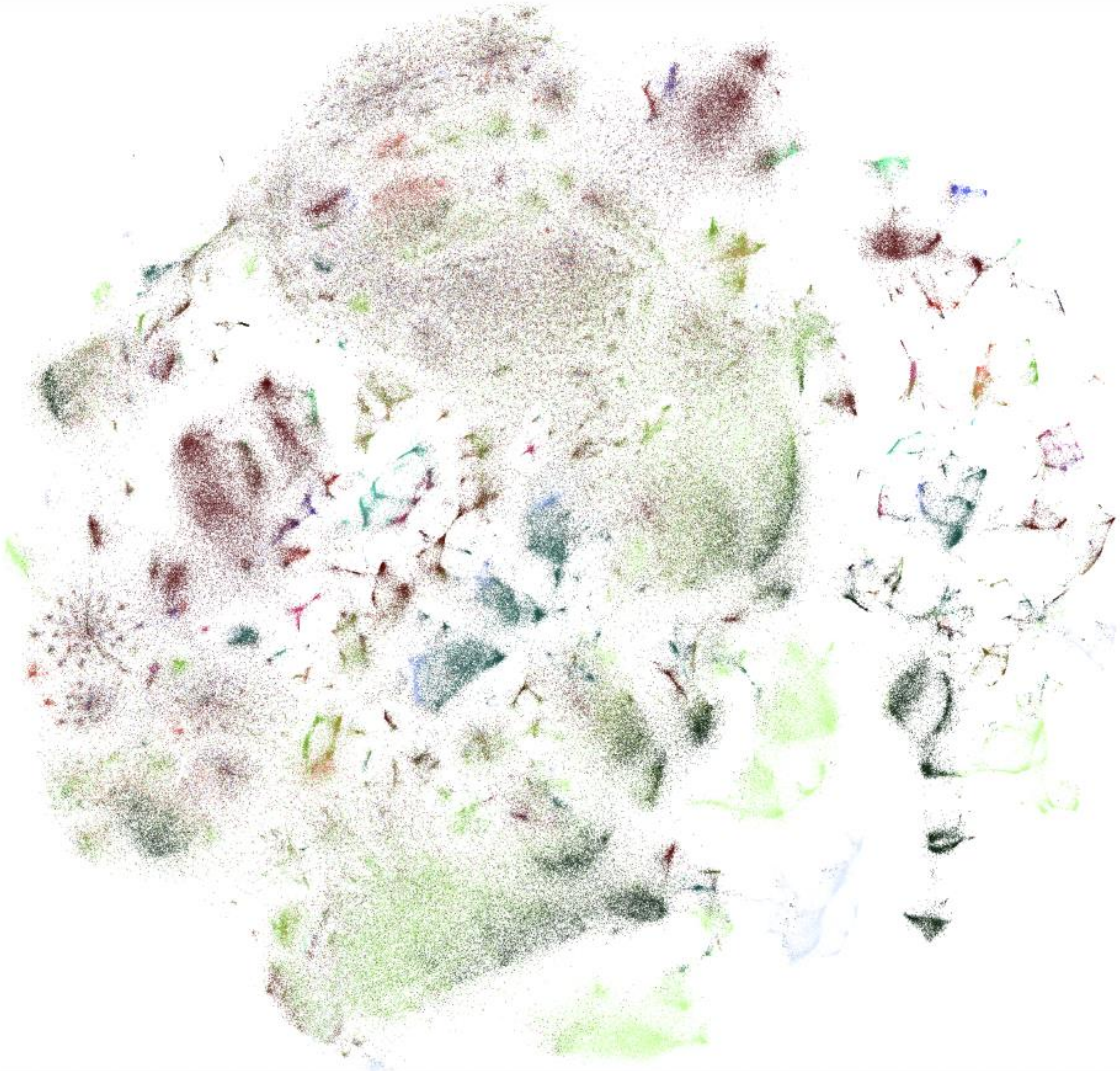
Open-sourced github repository, code for this figure: https://github.com/linnarsson-lab/adult-human-brain/blob/main/notebooks/Preprint/Figure1.ipynb

Million-level cell embedding Visualization

Alternative Visualization Published work based on PCA + t-SNE

We firstly adopt PCA for 200 dimensions, then use the t-SNE algorithm for final dimension reduction. We find the result is good, but the clusters are still not so perfectly separated.





453k cell embedding vis:

- PCA, dimension to 200
- T-SNE, dimension to 2 (scanpy default T-SNE)
- Two-step normalizaiton

Million-level cell embedding Visualization

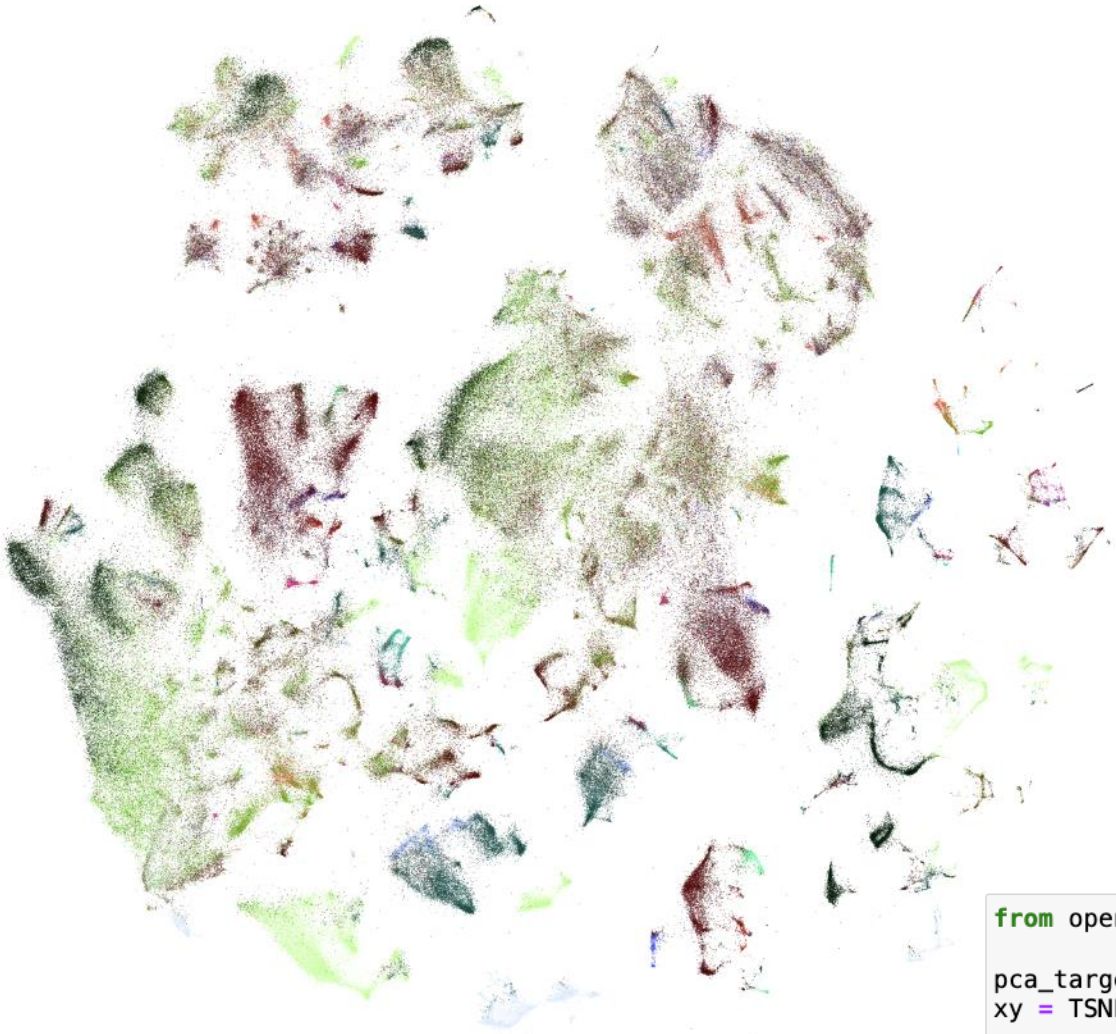
Alternative Visualization Published work based on PCA + t-SNE

We then try the openTSNE (paralleled version) to obtain more t-SNE algorithm parameter settings.

```
sc.tl.pca(adata_concat, n_comps=200)
```

```
from openTSNE import TSNE

pca_target = adata_concat.obsm['X_pca']
xy = TSNE(perplexity=20, exaggeration=1.5).fit(pca_target)
```

```
from openTSNE import TSNE

pca_target = adata_concat.obsm['X_pca']
xy = TSNE(perplexity=20, exaggeration=2.8).fit(pca_target)
```

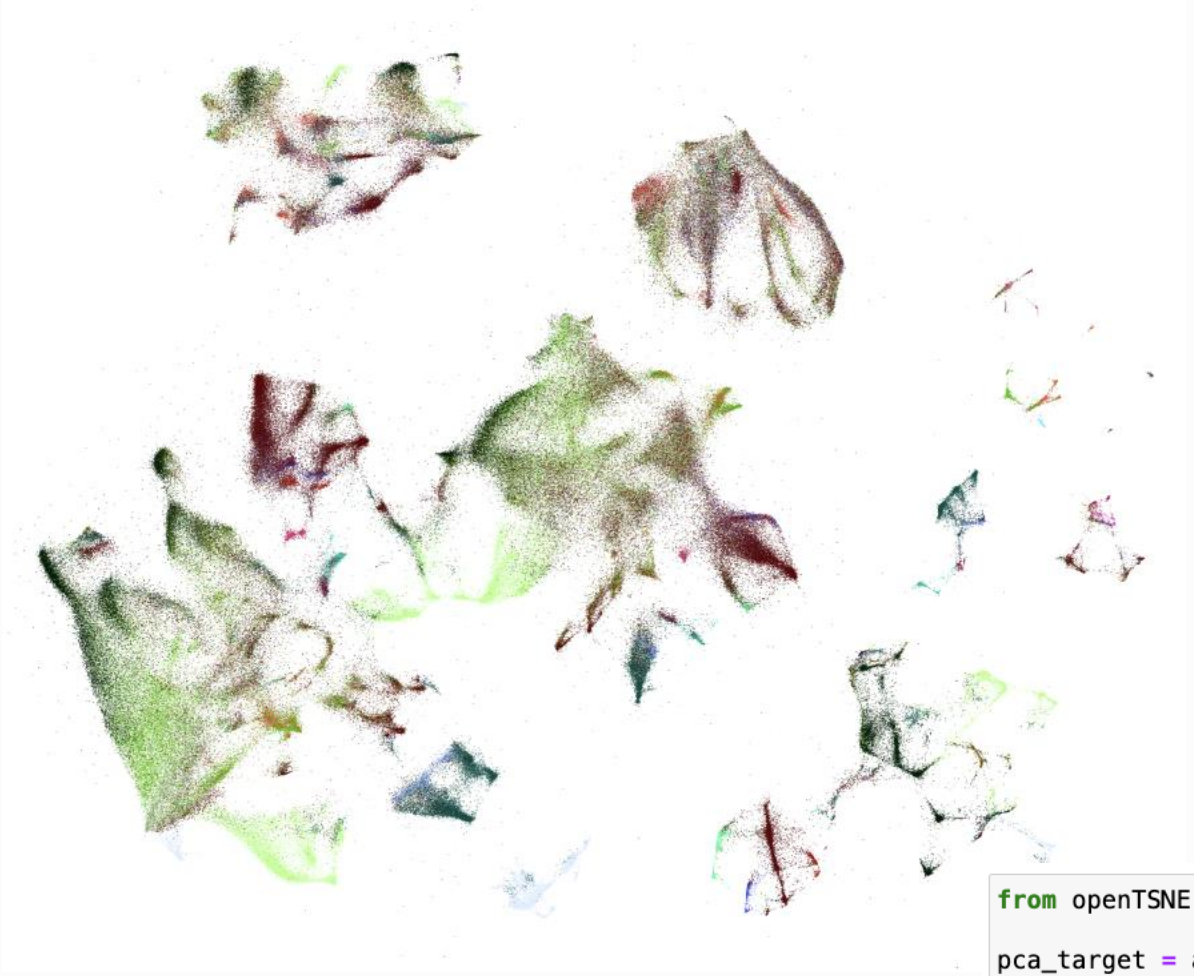


453k cell embedding vis:

- PCA, dimension to 200
- T-SNE, dimension to 2 (scanpy default T-SNE)
- Two-step normalizaiton

T-SNE: openTSNE package

```
from openTSNE import TSNE

pca_target = adata_concat.obsm['X_pca']
xy = TSNE(perplexity=20, exaggeration=1.5).fit(pca_target)
```



453k cell embedding vis:

- PCA, dimension to 200
- T-SNE, dimension to 2 (scanpy default T-SNE)
- Two-step normalizaiton

T-SNE: openTSNE package

```
from openTSNE import TSNE

pca_target = adata_concat.obsm['X_pca']
xy = TSNE(perplexity=20, exaggeration=2.8).fit(pca_target)
```

Million-level cell embedding Visualization

Alternative Visualization Published work based on PCA + t-SNE

We then try the openTSNE (paralleled version) to obtain more t-SNE algorithm parameter settings.

```
sc.tl.tsne(adata_concat, use_rep='X_pca')
```

```
from openTSNE import TSNE

pca_target = adata_concat.obsm['X_pca']
xy = TSNE(perplexity=20, exaggeration=1.5).fit(pca_target)
```

```
from openTSNE import TSNE

pca_target = adata_concat.obsm['X_pca']
xy = TSNE(perplexity=20, exaggeration=2.8).fit(pca_target)
```

As the ***exaggeration*** rate is higher, the clusters are more clearly separated in the visualization.

Million-level cell embedding Visualization

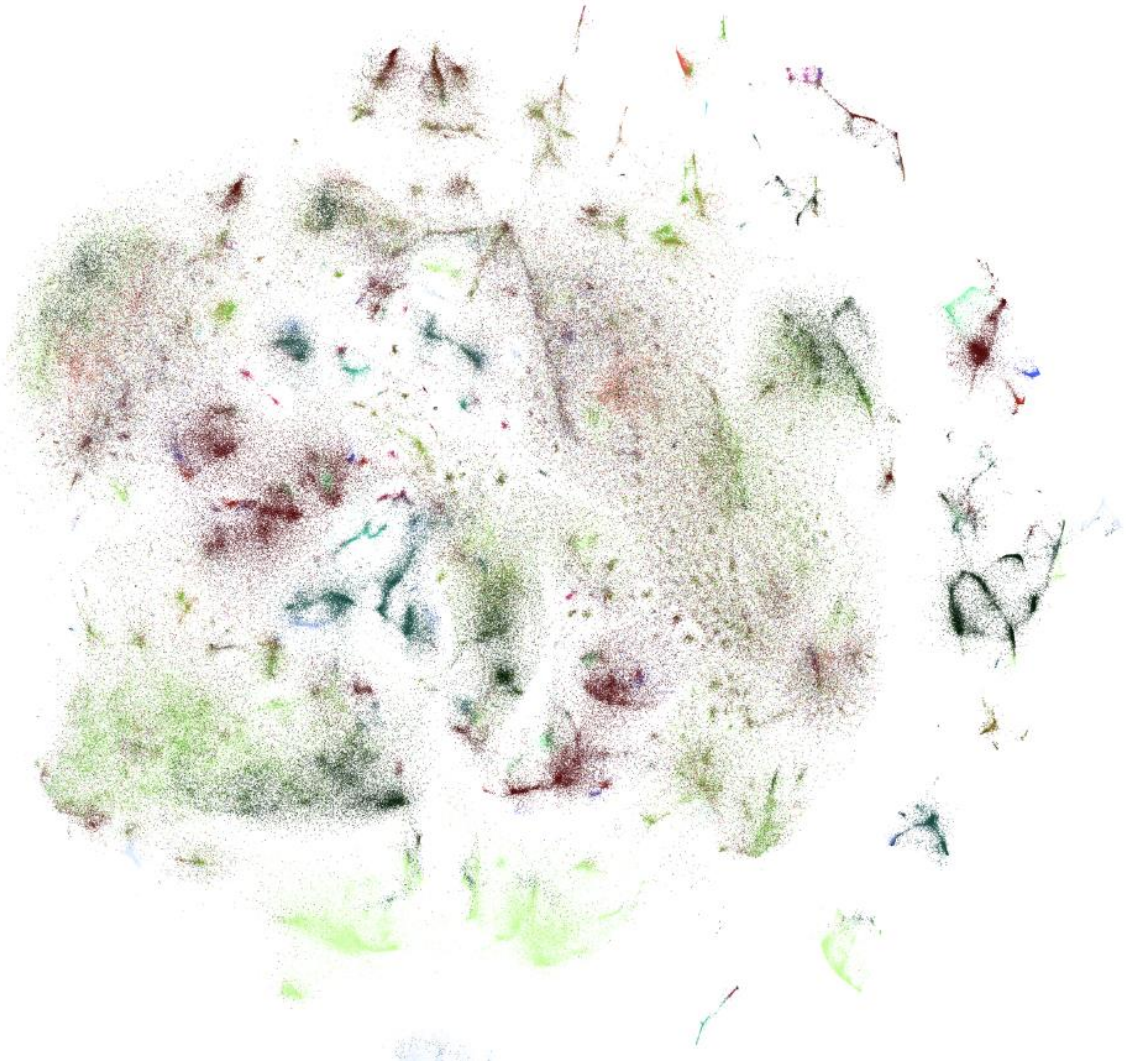
Alternative Visualization Published work based on PCA + t-SNE

To avoid the bias introduced by PCA re-processing dimension settings, we test 1000 dimension setting. Theoretically, the higher dimension number chosen should reflect better original embedding information.

```
sc.tl.pca(adata_concat, n_comps=1000)
```

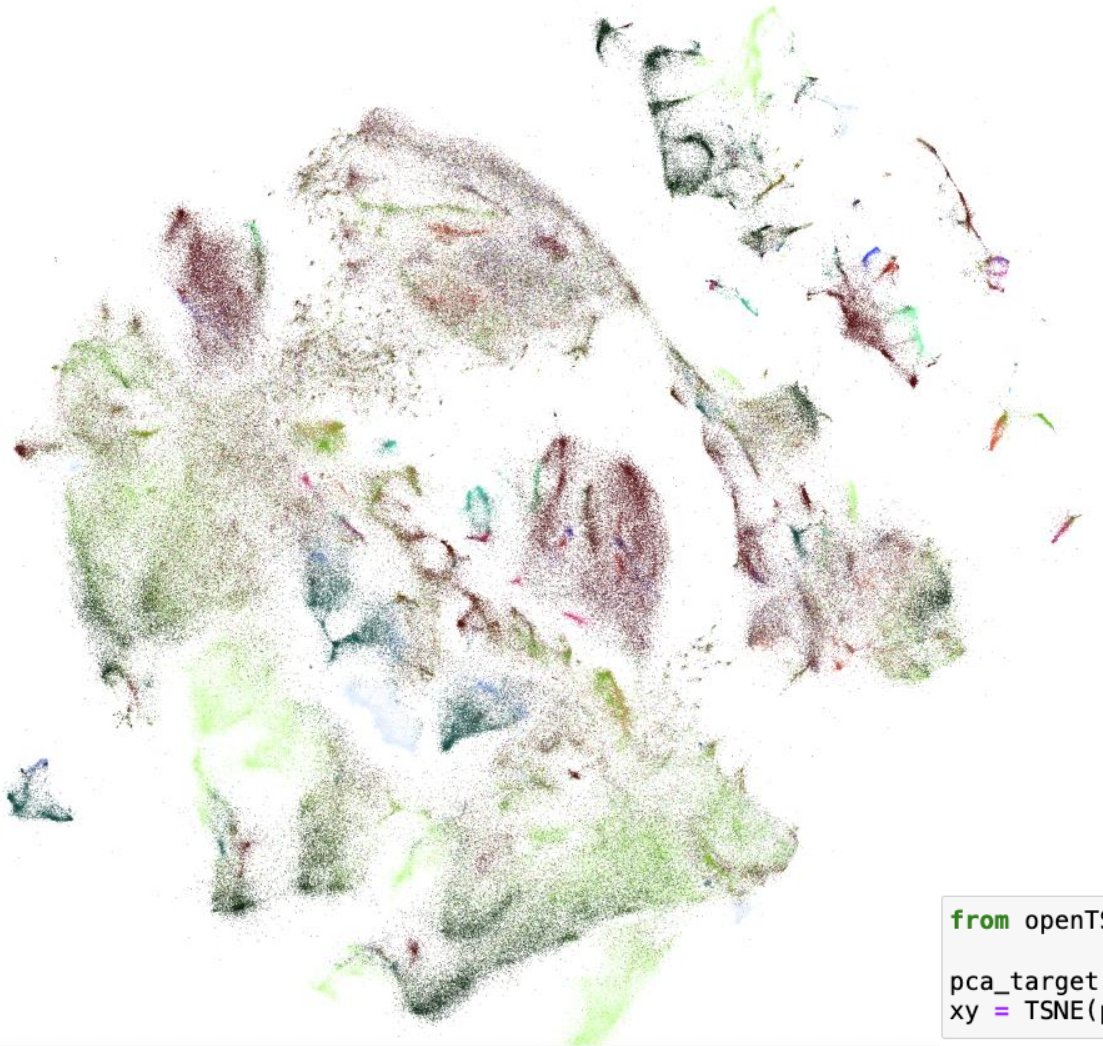
```
from openTSNE import TSNE

pca_target = adata_concat.obsm['X_pca']
xy = TSNE(perplexity=20, exaggeration=1.5).fit(pca_target)
```

```
from openTSNE import TSNE

pca_target = adata_concat.obsm['X_pca']
xy = TSNE(perplexity=20, exaggeration=2.8).fit(pca_target)
```



453k cell embedding vis:

- PCA, dimension to **1000**
- T-SNE, dimension to 2 (scanpy default T-SNE)
- Two-step normalizaiton



453k cell embedding vis:

- PCA, dimension to **1000**
- T-SNE, dimension to 2 (scanpy default T-SNE)
- Two-step normalizaiton

T-SNE: openTSNE package

```
from openTSNE import TSNE

pca_target = adata_concat.obsm['X_pca']
xy = TSNE(perplexity=20, exaggeration=1.5).fit(pca_target)
```

453k cell embedding vis:

- PCA, dimension to **1000**
- T-SNE, dimension to 2 (scanpy default T-SNE)
- Two-step normalizaiton

T-SNE: openTSNE package

```
from openTSNE import TSNE

pca_target = adata_concat.obsm['X_pca']
xy = TSNE(perplexity=20, exaggeration=2.8).fit(pca_target)
```

Million-level cell embedding Visualization

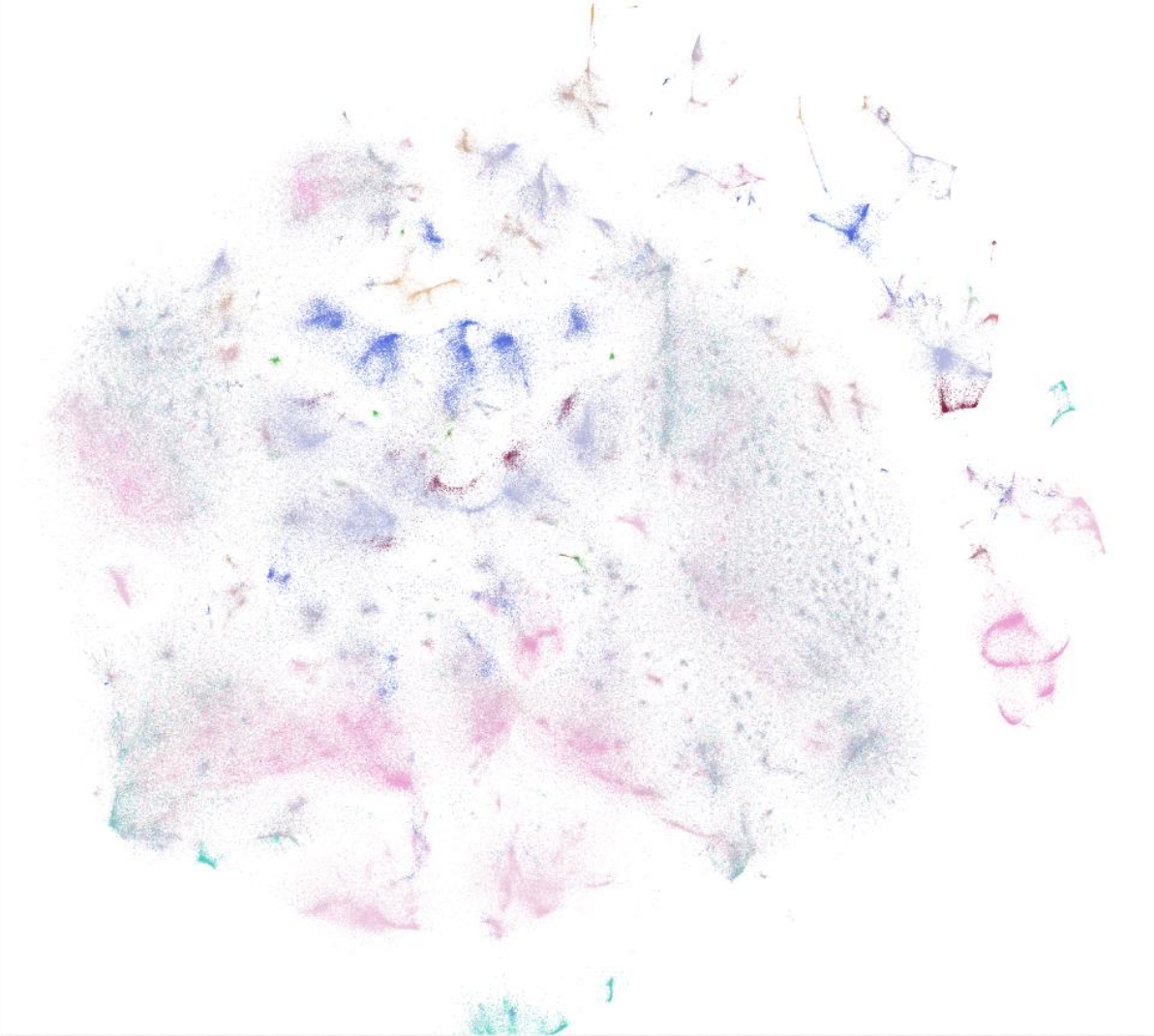
Alternative Visualization Published work based on PCA + t-SNE

For 1000 PCA setting, to better understand the most prevalent cell types, we filtered out the cell categories that is less than 1k in the whole embedding. *453k → 448k*

```
sc.tl.pca(adata_concat, n_comps=1000)
```

```
from openTSNE import TSNE

pca_target = adata_concat.obsm['X_pca']
xy = TSNE(perplexity=20, exaggeration=1.5).fit(pca_target)
```

```
from openTSNE import TSNE

pca_target = adata_concat.obsm['X_pca']
xy = TSNE(perplexity=20, exaggeration=2.8).fit(pca_target)
```



453k cell embedding vis:

- PCA, dimension to **1000**
- T-SNE, dimension to 2 (scanpy default T-SNE)
- Two-step normalizaiton
- Filtered cells that less than 1k of its type

453k cell embedding vis:

- PCA, dimension to **1000**
- T-SNE, dimension to 2 (scanpy default T-SNE)
- Two-step normalizaiton
- Filtered cells that less than 1k of its type

T-SNE: openTSNE package

```
from openTSNE import TSNE

pca_target = adata_concat.obsm['X_pca']
xy = TSNE(perplexity=20, exaggeration=1.5).fit(pca_target)
```

453k cell embedding vis:

- PCA, dimension to **1000**
- T-SNE, dimension to 2 (scanpy default T-SNE)
- Two-step normalizaiton
- Filtered cells that less than 1k of its type

T-SNE: openTSNE package

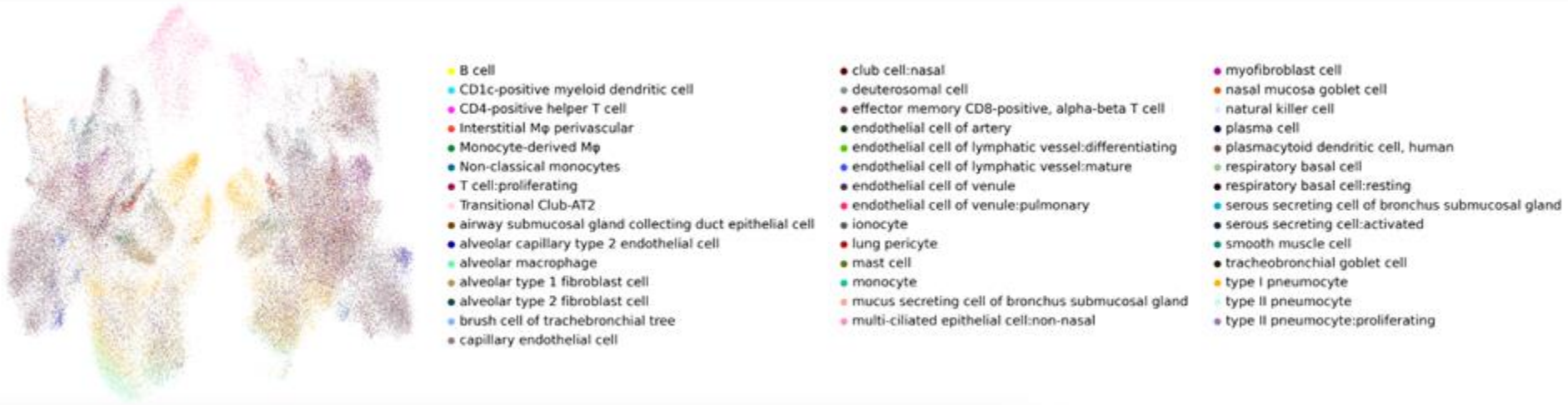
```
from openTSNE import TSNE

pca_target = adata_concat.obsm['X_pca']
xy = TSNE(perplexity=20, exaggeration=2.8).fit(pca_target)
```

450K cell embedding vis

[Correct embedding layer + Normalization]: HBM975.WQQQ.853

Special thanks to Andi, Bruce, Daniel, Yash and Filipi's suggestions

GitHub repo: https://github.com/cns-iu/hra-cell-embeddings



Special settings:

1. Utilize the correct matrix layer:

```
'spliced_unspliced_sum'
```

2. Adopt two steps of normalization before conducting the UMAP function.

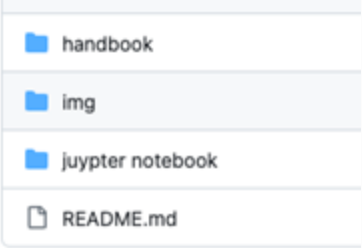
```
sc.pp.normalize_total(adata)
sc.pp.log1p(adata)
```

Concatenation of two datasets: in total 202k cells

hubmap id
HBM948.GXMD.986
HBM975.WQQQ.853

Different concatenated dataset selections would lead to different visualization effects.

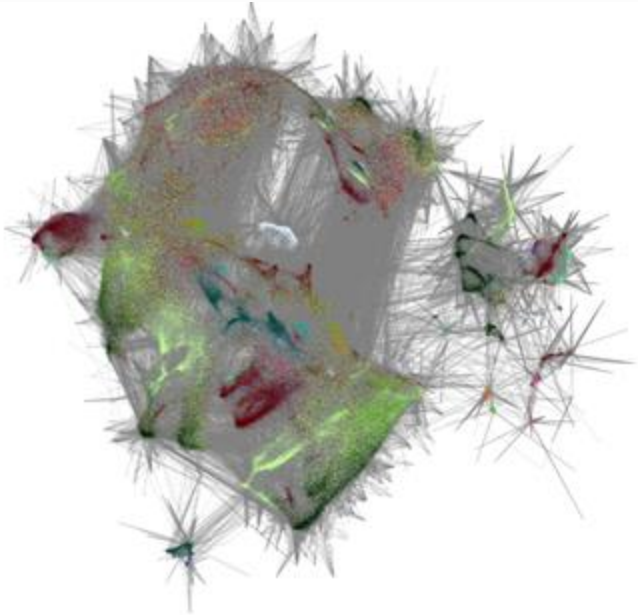
450K cell embedding vis



Github Repo:
https://github.com/cns-iu/hra-cell-embeddings

Visualization for all 453k cells. (t-SNE)

2. Visualization Tuning - PCA-200 + tSNE / Visualization Tuning - PCA-1000 + tSNE: we adopt the PCA algorithm for 200/1000 dimensions and then for t-SNE algorithm, with the same normalization settings in the Basic Visualization jupyter notebook. The visualization for the PCA setting of 1000 is shown below:



Suggestion from Filipi:

- The current settings of current two step normalization is fine, more configuration information about the pre-process of those datasets would be helpful to further improve

UMAP was supposed to work well in this data, however it is underperforming compared to TSNE

450K cell embedding vis

Exclude all the potential errors making the previous visualisation not so perfectly clustered — with the UMAP approach

Normalization 10 datasets in same settings, UMAP after 1000 dimension of PCA pre-processing.

Using not trained model

Normalization 10 datasets in same settings, UMAP after 1000 dimension of PCA pre-processing.

Using trained model

450K cell embedding vis

Different normalization settings, only log1p.
— The clusters are too separated.

```python
from scarches.models.scpoli._utils import reads_to_fragments
adata_fragments = reads_to_fragments(adata_concat, copy=True)
adata_fragments

from scarches.models.scpoli import scPoli

scpoli_model = scPoli(
    adata=adata_fragments,
    condition_keys=condition_key,
    cell_type_keys=cell_type_key,
    hidden_layer_sizes=[100],
    latent_dim=25,
    embedding_dims=5,
    recon_loss='poisson',
)

scpoli_model.train(
    n_epochs=100
)
```

Normalization 10 datasets in same settings, UMAP after 1000 dimension of PCA pre-processing.

Using not trained model

Normalization 10 datasets in same settings, UMAP after 1000 dimension of PCA pre-processing.

Using trained model

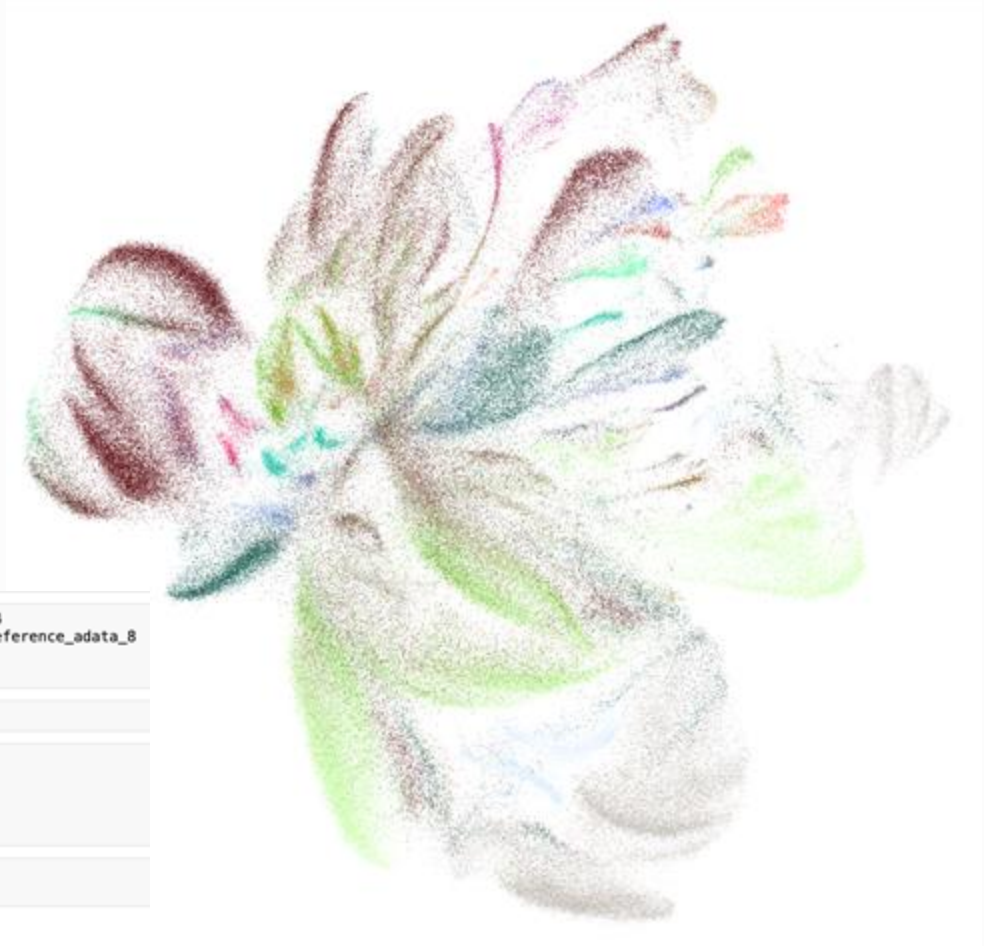
450K cell embedding vis

Post normalization: *concat then do the normalization*

Normalization 10 datasets in same settings, UMAP after 1000 dimension of PCA pre-processing.

Using not trained model (which it should be)

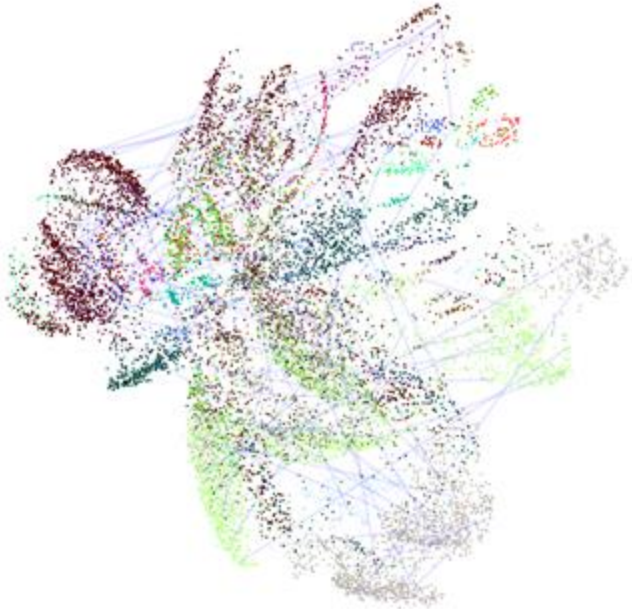
```
In [9]: ann_data_list = [reference_adata_1, reference_adata_2, reference_adata_3, reference_adata_4
                                , reference_adata_5, reference_adata_6, reference_adata_7, reference_adata_8
                                , reference_adata_9, reference_adata_10]
        # ann_data_list

In [10]: combined_ann_data = ann_data_list[0].concatenate(ann_data_list[1:])

In [11]: sc.pp.filter_cells(combined_ann_data, min_genes=200)
         sc.pp.filter_genes(combined_ann_data, min_cells=3)
         sc.pp.normalize_total(combined_ann_data, target_sum=1e4)
         sc.pp.log1p(combined_ann_data)
         sc.pp.highly_variable_genes(combined_ann_data, n_top_genes=2000)

In [12]: adata_concat = combined_ann_data
         adata_concat

Out[12]: AnnData object with n_obs × n_vars = 259531 × 53221
             obs: 'CL_label', 'batch', 'n_genes'
             var: 'hugo_symbol', 'n_cells', 'highly_variable', 'means', 'dispersions', 'dispersions_norm'
             uns: 'log1p', 'hvg'
             layers: 'spliced', 'spliced_unspliced_sum', 'unspliced'
```



450K cell embedding vis

Error analysis

1. Sample 5% of all nodes
2. Randomly connect a fraction of pairs (0.001%) that belong to the same category
3. Visualize



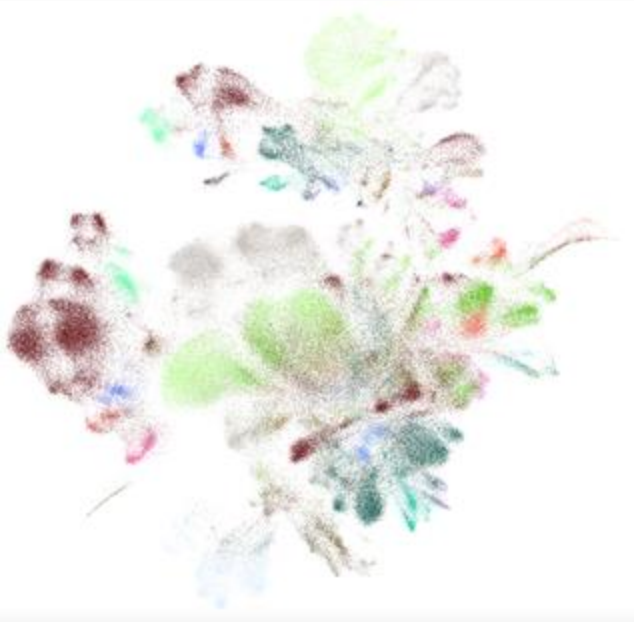
- Alveolar Mφ proliferating
- B cell
- CD1c-positive myeloid dendritic cell
- CD4-positive helper T cell
- Interstitial Mφ perivascular
- Monocyte-derived Mφ
- Non-classical monocytes
- T cell:proliferating
- Transitional Club-AT2
- airway submucosal gland collecting duct epithelial cell
- alveolar capillary type 2 endothelial cell
- alveolar macrophage
- alveolar type 1 fibroblast cell
- alveolar type 2 fibroblast cell
- brush cell of trachebronchial tree
- capillary endothelial cell
- club cell:nasal
- club cell:non-nasal
- deuterosomal cell
- effector memory CD8-positive, alpha-beta T cell
- endothelial cell of artery
- endothelial cell of lymphatic vessel:differentiating
- endothelial cell of lymphatic vessel:mature
- endothelial cell of venule
- endothelial cell of venule:pulmonary
- ionocyte
- lung pericyte
- mast cell
- monocyte
- mucus secreting cell of bronchus submucosal gland
- multi-ciliated epithelial cell:nasal
- multi-ciliated epithelial cell:non-nasal
- myofibroblast cell
- nasal mucosa goblet cell
- natural killer cell
- plasma cell
- plasmacytoid dendritic cell, human
- pulmonary interstitial fibroblast
- respiratory basal cell
- respiratory basal cell:resting
- serous secreting cell of bronchus submucosal gland
- serous secreting cell:activated
- smooth muscle cell
- tracheobronchial goblet cell
- type I pneumocyte
- type II pneumocyte
- type II pneumocyte:proliferating

450K cell embedding vis

Use trained model to improve the clustering

Original

hidden_layer_size=100



hidden_layer_size=1000

450K cell embedding vis

Utilize the UCE model to improve the cell embedding before UMAP

Previous UMAP

4-layer UCE (1.5 hours)

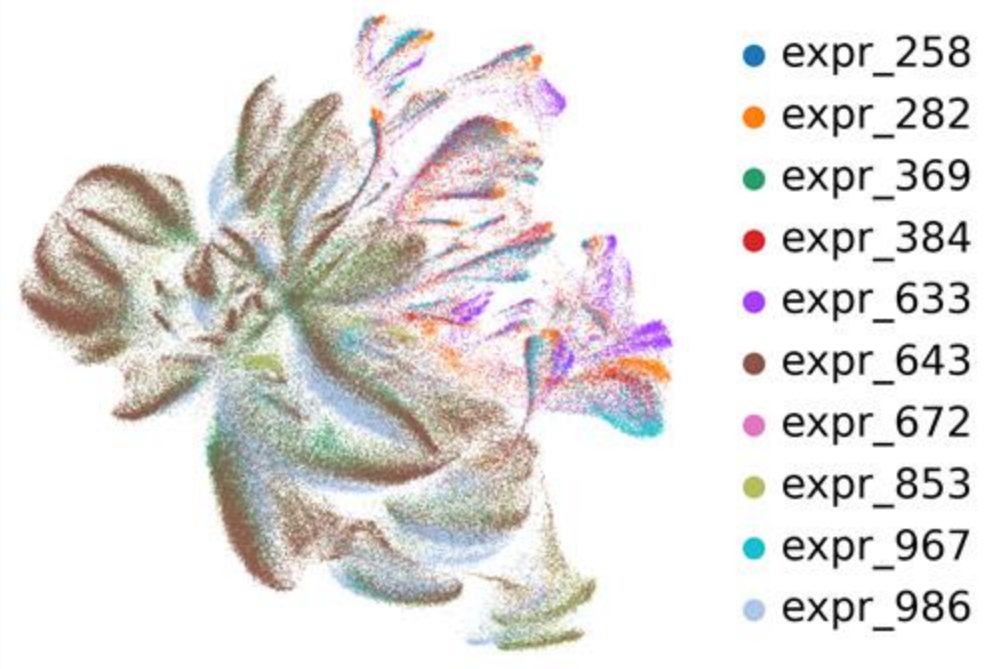
33-layer UCE (7 hours)

450K cell embedding vis

Visualization colored by the source dataset information.

Vis [baseline setting via UMAP]

Visualization colored by source datasets information

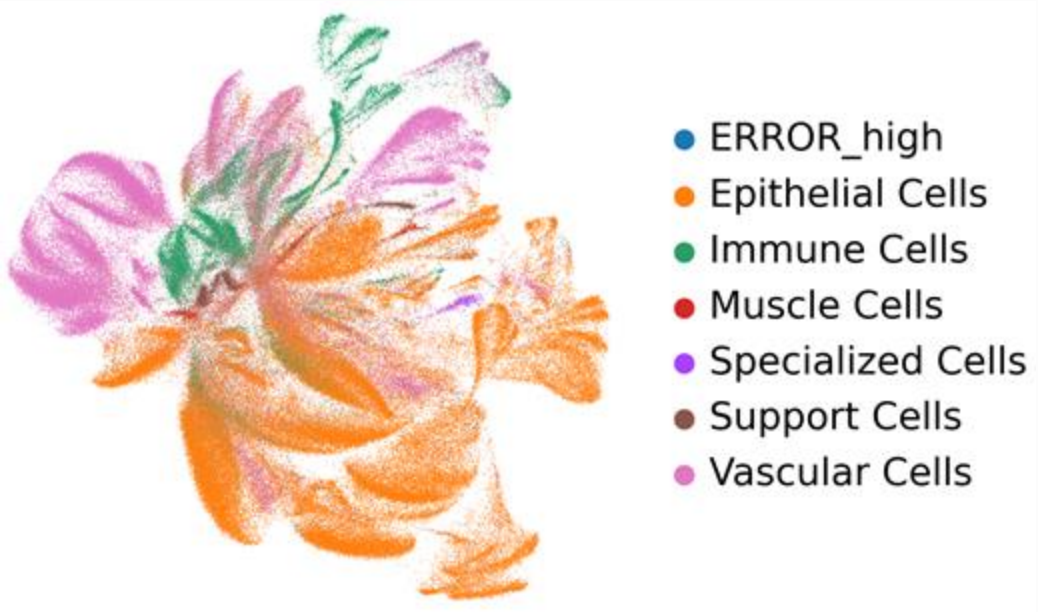


450K cell embedding vis

Visualization colored by high level cell type information.

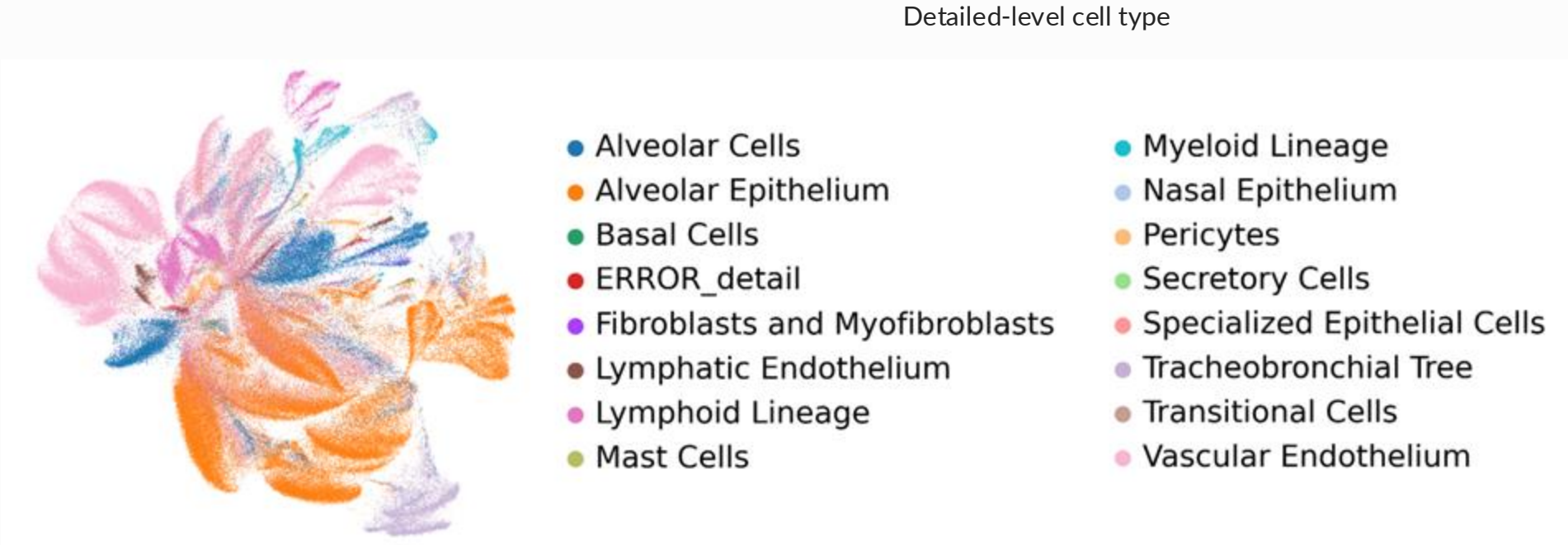
Vis [baseline setting via UMAP]

High-level cell type



450K cell embedding vis

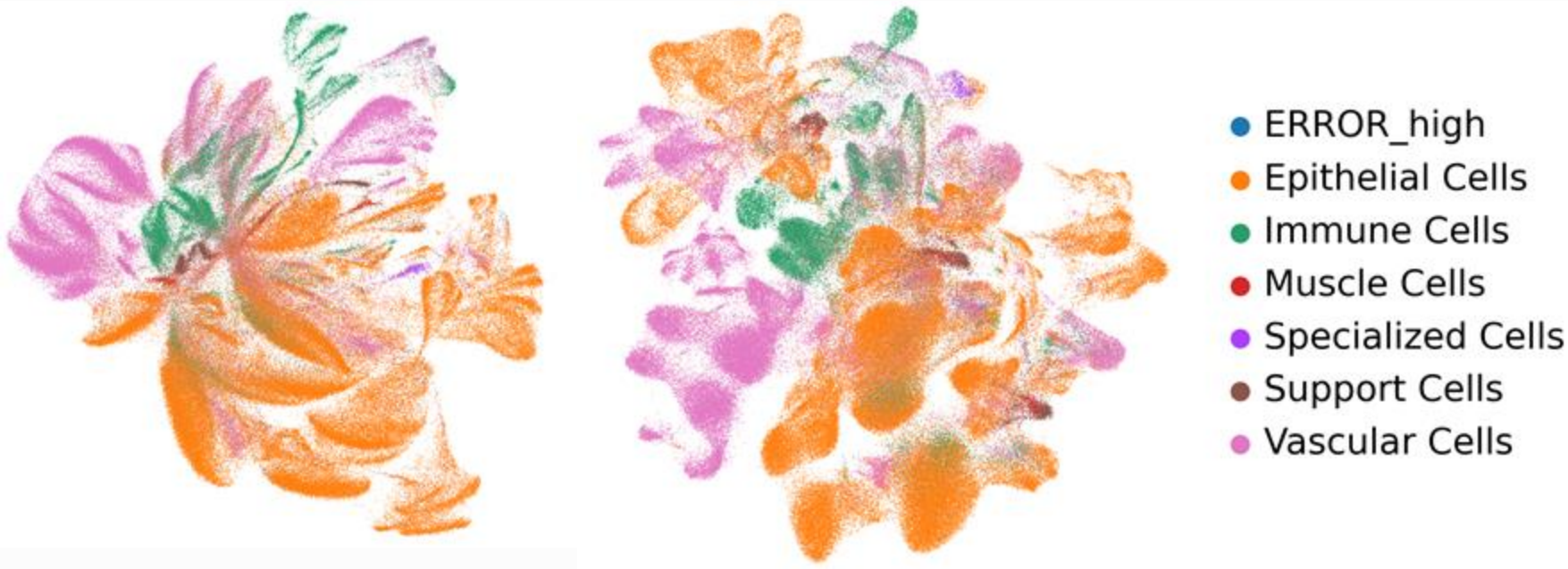
Visualization colored by relatively detailed level cell type information.



450K cell embedding vis

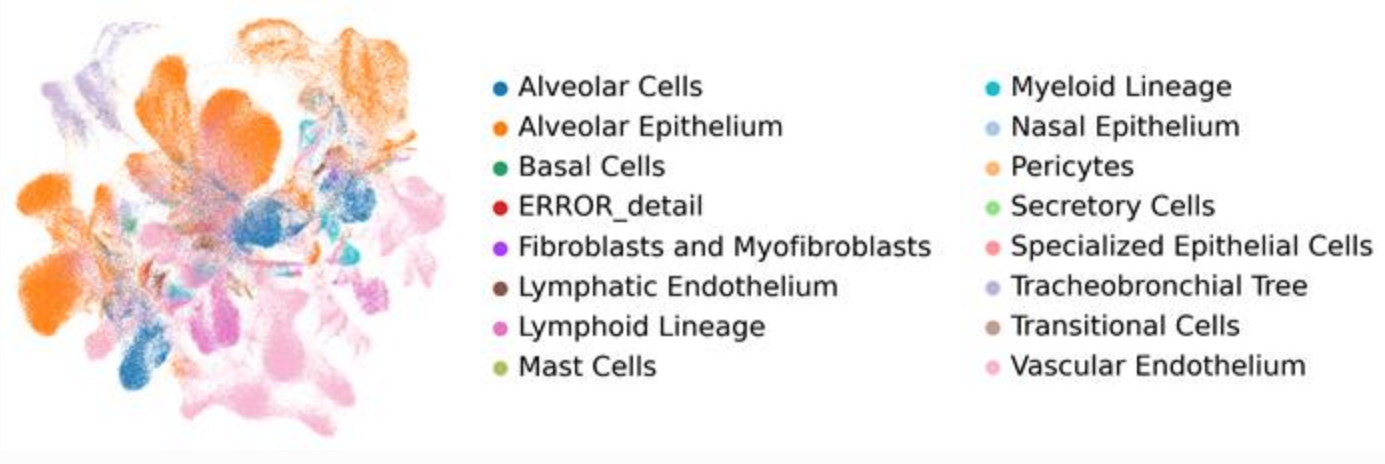
Training attempts on 10 lung datasets, based on [**high_level**, detailed_level]
on [**original cell embeddings**, UCE_4, UCE_33], then troubleshooting (based on edges).

High-level cell type



450K cell embedding vis

Training attempts on 10 lung datasets, based on [high_level, **detailed_level**]
on [**original cell embeddings**, UCE_4, UCE_33], then troubleshooting (based on edges).



450K cell embedding vis

Training attempts on 10 lung datasets, based on [**high_level**, detailed_level]
on [original cell embeddings, **UCE_4**, UCE_33], then troubleshooting (based on edges).

Not-trained & original embedding

Trained on high_level & original embedding

Trained on high_level & UCE_4

450K cell embedding vis

Training attempts on 10 lung datasets, based on [high_level, **detailed_level**]
on [original cell embeddings, **UCE_4**, UCE_33], then troubleshooting (based on edges).

Not-trained & original embedding

Trained on detailed_level & original embedding

Trained on detailed_level & UCE_4

450K cell embedding vis

Training attempts on 10 lung datasets, based on [**high_level**, detailed_level]
on [original cell embeddings, UCE_4, **UCE_33**], then troubleshooting (based on edges).

Not-trained & original embedding

Trained on high_level & UCE_4

Trained on high_level & UCE_33

450K cell embedding vis

Training attempts on 10 lung datasets, based on [high_level, **detailed_level**]
on [original cell embeddings, UCE_4, **UCE_33**], then troubleshooting (based on edges).

Not-trained & original embedding

Trained on detailed_level & UCE_4

Trained on detailed_level & UCE_33

450K cell embedding vis

Trying out the GTEx dataset visualization, in paper:
https://www.science.org/doi/10.1126/science.abl4290

Their UMAP

Their PCA

Our setting default PCA before UMAP

450K cell embedding vis

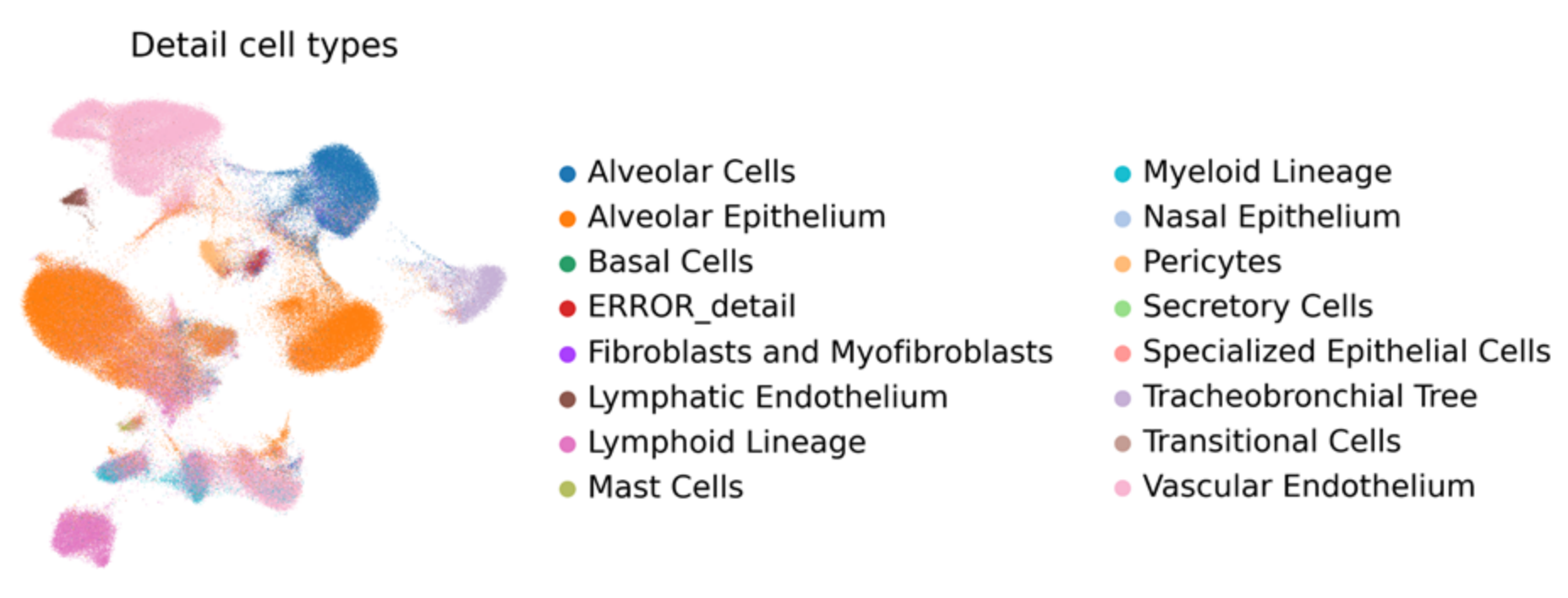
GTEx Visualization

Replication Settings on:

1. sc.pp.highly_variable_genes
2. harmonize(adata_concat.obsm['X_pca'], adata_concat.obs, **batch_key**)
   * ***Bulk-peudobulk settings***

Apply the settings learnt in GTEx replications.

450K cell embedding vis



Visualization Conlcusion

Different normalization setting on trained models

not trained, 2 step normalization

trained, 1 step normalization

trained, 2 step normalization

Visualization Conlcusion

Utilize the UCE model to improve the clustering effect of the cell embeddings before UMAP

Basic UMAP

4-layer UCE (1.5 hours)

33-layer UCE (7 hours)

Visualization Conlcusion

Utilize the detailed cell type categorization settings

Basic UMAP

Trained, UCE 33-layer model

GTEx-style clustering