In [1]:
```python
import numpy as np
import pandas as pd
import scanpy as sc

import matplotlib as mpl
import matplotlib.pyplot as plt
from scipy import sparse, io
%matplotlib inline

import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
warnings.simplefilter(action='ignore', category=UserWarning)

from scipy.sparse import csr_matrix
```

In [2]:
```python
sc.logging.print_header()
```

```
scanpy==1.8.2 anndata==0.7.6 umap==0.5.1 numpy==1.21.5 scipy==1.7.0 pandas==1.4.1 scikit-learn==0.24.2 statsmod
els==0.12.2 python-igraph==0.9.9 louvain==0.7.1 pynndescent==0.5.4
```

In [3]:
```python
sc.settings.verbosity = 3             # verbosity: errors (0), warnings (1), info (2), hints (3)
sc.settings.set_figure_params(dpi=150, facecolor='white')

from IPython.display import HTML
HTML("<style>.container { width:100% !important; }</style>")
```

Out[3]:

Prepare query data

Preprocessed developmental mouse midbrain brain data from the paper La Manno, G., Siletti, K., Furlan, A. et al. Molecular architecture of the developing mouse brain. Nature 596, 92–96 (2021). https://doi.org/10.1038/s41586-021-03775-x

In [4]:
```python
adata_query = sc.read_h5ad('../adata_timepoints.h5ad')
```

In [5]:
```python
adata_query
```

Out[5]:
```
AnnData object with n_obs × n_vars = 83703 × 23901
    obs: 'batch', 'n_genes', 'n_genes_by_counts', 'total_counts', 'total_counts_mt', 'pct_counts_mt', 'SampleI
D', 'Tissue', 'Age'
    var: 'gene_name', 'gene_id', 'n_cells', 'mt', 'n_cells_by_counts', 'mean_counts', 'pct_dropout_by_counts',
'total_counts'
    layers: 'spliced', 'unspliced'
```

In [6]:
```python
adata_query.obs['Tissue'].cat.categories
```

Out[6]:
```
Index(['All (No extraembryonal)', 'Cortex ', 'Head', 'Midbrain',
       'MidbrainDorsal', 'MidbrainVentral'],
      dtype='object')
```

In [7]:
```python
adata_query.obs['Age'].cat.categories
```

Out[7]:
```
Index(['e7.0', 'e8.0', 'e8.5', 'e9.0', 'e10.0', 'e11.0', 'e11.5', 'e12.0',
       'e12.5', 'e13.5', 'e14.0', 'e14.5', 'e15.0', 'e15.5', 'e16.0', 'e16.5',
       'e17.5', 'e18.0', 'p0', 'p05', 'p07'],
      dtype='object')
```

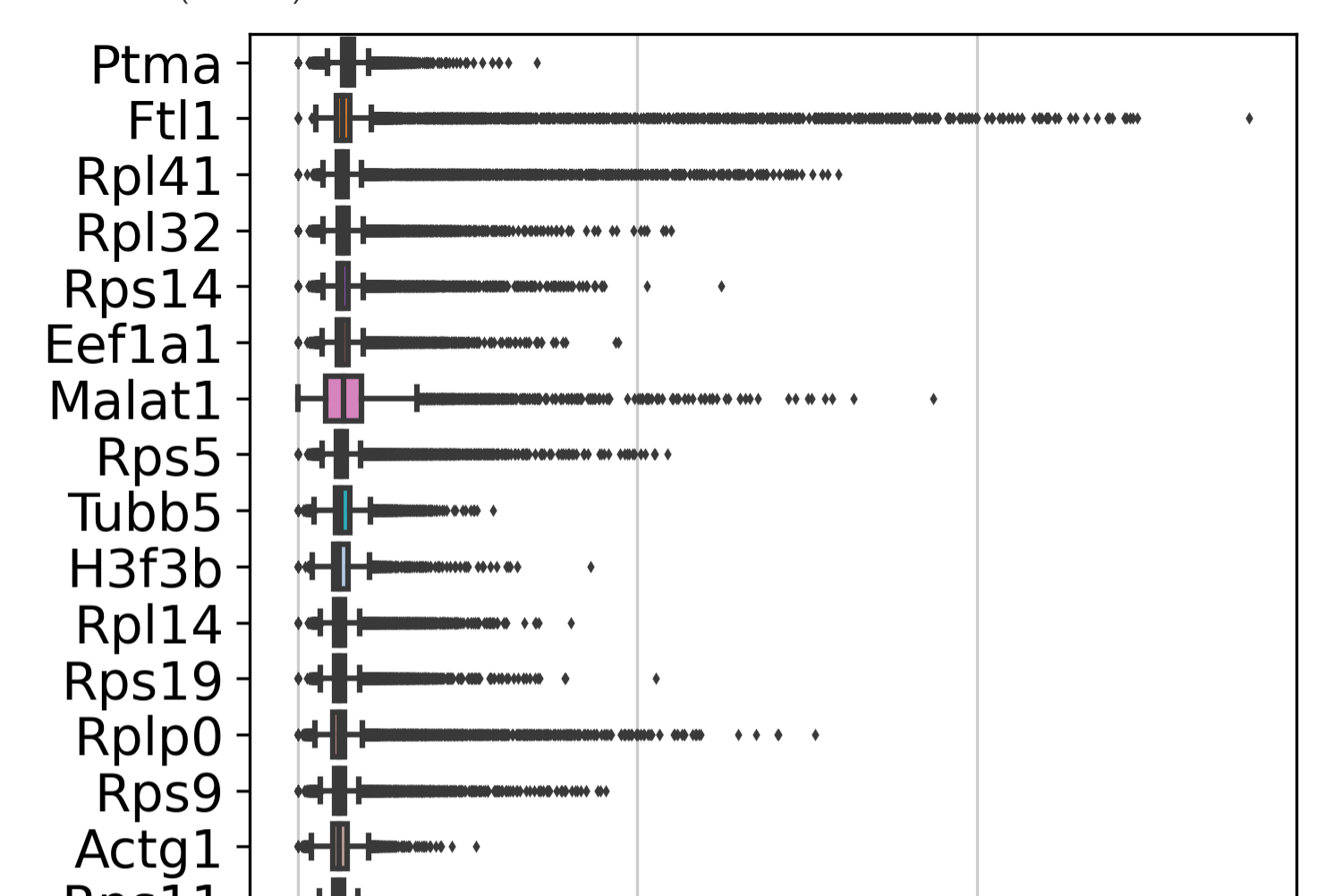
In [8]:
```python
adata_query.layers["counts"] = adata_query.X.copy() # preserve counts
```

In [9]:
```python
# Total-count normalize (library-size correct) the data matrix X to 10000 reads per cell,
# so that counts become comparable among cells.
# It will eliminate cell-specific biases resulted from capture efficiency, sequencing depth, dropouts,
# and other technical effects.
sc.pp.normalize_total(adata_query, target_sum=1e4) # scale each cell to a common library size
sc.pp.log1p(adata_query) # log(expression + 1)
```

```
normalizing counts per cell
    finished (0:00:00)
```

In [10]:
```python
sc.pl.highest_expr_genes(adata_query, n_top=20)
```

```
normalizing counts per cell
    finished (0:00:01)
```



In [11]:
```python
adata_query.raw = adata_query # freeze the state in `.raw`
```

In [12]:
```python
sc.pp.highly_variable_genes(
    adata_query,
    n_top_genes=5000,
    # subset=True, # to automatically subset to the 5000 genes
    layer="counts",
    flavor="seurat_v3"
)
```

```
If you pass `n_top_genes`, all cutoffs are ignored.
extracting highly variable genes
--> added
    'highly_variable', boolean vector (adata.var)
    'highly_variable_rank', float vector (adata.var)
    'means', float vector (adata.var)
    'variances', float vector (adata.var)
    'variances_norm', float vector (adata.var)
```

In [13]:
```python
adata_query
```

Out[13]:
```
AnnData object with n_obs × n_vars = 83703 × 23901
    obs: 'batch', 'n_genes', 'n_genes_by_counts', 'total_counts', 'total_counts_mt', 'pct_counts_mt', 'SampleI
D', 'Tissue', 'Age'
    var: 'gene_name', 'gene_id', 'n_cells', 'mt', 'n_cells_by_counts', 'mean_counts', 'pct_dropout_by_counts',
'total_counts', 'highly_variable', 'highly_variable_rank', 'means', 'variances', 'variances_norm'
    uns: 'log1p', 'hvg'
    layers: 'spliced', 'unspliced', 'counts'
```

Prepare reference dataset

Allen Brain Map - Mouse Whole Cortex and Hippocampus from ~8 weeks old mice

In [14]:
```python
adata_ref= sc.read_h5ad('../mouse_subset.h5ad')
```

In [15]:
```python
adata_ref
```

Out[15]:
```
AnnData object with n_obs × n_vars = 199985 × 31053
    obs: 'batch', 'donor_sex_label', 'cell_type'
    var: 'gene_name'
    obsm: 'X_umap'
```

In [16]:
```python
adata_ref.obs['cell_type'].cat.categories
```

Out[16]:
```
Index(['GABAergic-Lamp5', 'GABAergic-Meis2', 'GABAergic-Pvalb',
       'GABAergic-Sncg', 'GABAergic-Sst', 'GABAergic-Sst Chodl',
       'GABAergic-Vip', 'Glutamatergic-CA2-IG-FC', 'Glutamatergic-CR',
       'Glutamatergic-CT SUB', 'Glutamatergic-Car3',
       'Glutamatergic-L2 IT ENTl', 'Glutamatergic-L2/3 IT CTX',
       'Glutamatergic-L2/3 IT ENTl', 'Glutamatergic-L2/3 IT PPP',
       'Glutamatergic-L2/3 IT RHP', 'Glutamatergic-L4 RSP-ACA',
       'Glutamatergic-L4/5 IT CTX', 'Glutamatergic-L5 IT CTX',
       'Glutamatergic-L5 PT CTX', 'Glutamatergic-L5/6 IT TPE-ENT',
       'Glutamatergic-L5/6 NP CTX', 'Glutamatergic-L6 CT CTX',
       'Glutamatergic-L6 IT CTX', 'Glutamatergic-L6 IT ENTl',
       'Glutamatergic-L6b CTX', 'Glutamatergic-L6b/CT ENT',
       'Glutamatergic-NP PPP', 'Glutamatergic-NP SUB', 'Non-Neuronal-Astro',
       'Non-Neuronal-Endo', 'Non-Neuronal-Micro-PVM', 'Non-Neuronal-Oligo',
       'Non-Neuronal-SMC-Peri', 'Non-Neuronal-VLMC'],
      dtype='object')
```

In [17]:
```python
adata_ref.layers["counts"] = adata_ref.X.copy() # preserve counts
```

In [18]:
```python
sc.pp.highly_variable_genes(
    adata_ref,
    n_top_genes=5000,
#     subset=True,
    layer="counts",
    flavor="seurat_v3"
)
```

```
If you pass `n_top_genes`, all cutoffs are ignored.
extracting highly variable genes
--> added
    'highly_variable', boolean vector (adata.var)
    'highly_variable_rank', float vector (adata.var)
    'means', float vector (adata.var)
    'variances', float vector (adata.var)
    'variances_norm', float vector (adata.var)
```

Make a list with the highly variable genes from both query and reference datasets

Subset query and reference datasets based on the reunion of their highly variable genes

In [20]:
```python
hvg_query = []
for x in range(len(adata_query.var['highly_variable'])):
    if adata_query.var['highly_variable'][x] == True:
        hvg_query.append(adata_query.var['gene_name'][x])

hvg_ref = []
for x in range(len(adata_ref.var['highly_variable'])):
    if adata_ref.var['highly_variable'][x] == True:
        hvg_ref.append(adata_ref.var['gene_name'][x])
```

In [21]:
```python
hvg_sum = np.unique(hvg_query + hvg_ref)
len(hvg_sum)
```

Out[21]:
```
8041
```

In [22]:
```python
hvg_sum = list(set(set(hvg_sum) & set(adata_query.var.index) & set(adata_ref.var.index)))
```

In [23]:
```python
len(hvg_sum)
```

Out[23]:
```
6927
```

In [24]:
```python
adata_query = adata_query[:, hvg_sum].copy()
```

In [25]:
```python
adata_ref = adata_ref[:, hvg_sum].copy()
```

Workflow

1. Pretrain reference dataset with SCVI from scvi-tools

- scvi.model.SCVI.setup_anndata(adata_ref, batch_key="batch", layer="counts")
- vae_ref = scvi.model.SCVI(adata_ref, **arches_params)
- vae_ref.train(max_epochs = 100, use_gpu=True)

2. Train reference dataset with SCANVI from scvi-tools

- adata_ref.obs["labels_scanvi"] = adata_ref.obs["cell_type"].values ### treat all cells as being labeled
- vae_ref_scan = scvi.model.SCANVI.from_scvi_model(vae_ref, unlabeled_category="Unknown", labels_key="labels_scanvi")
- vae_ref_scan.train(max_epochs=20, use_gpu=True)

3. Transfer labels to query dataset

- adata_query.obs['labels_scanvi'] = vae_ref_scan.unlabeled*category* ### add unknown labels
- vae_q = scvi.model.SCANVI.load_query_data(adata_query, dir_path_scan)
- vae_q.train(max_epochs=50, plan_kwargs=dict(weight_decay=0.0), check_val_every_n_epoch=10, use_gpu=True)

_**weight_decay of 0.0** -> ensures that the latent representation of the reference cells will remain exactly the same if passing them through new query model._

In [26]:
```python
import scvi
```

```
Global seed set to 0
```

In [27]:
```python
print(scvi.__version__)
```

```
0.16.0
```

1. Pre-train reference with SCVI

In [28]:
```python
scvi.model.SCVI.setup_anndata(adata_ref, batch_key="batch", layer="counts")
```

In [29]:
```python
arches_params = dict(
    use_layer_norm="both",
    use_batch_norm="none",
    encode_covariates=True,
    dropout_rate=0.2,
    n_layers=2,
)

vae_ref = scvi.model.SCVI(
    adata_ref,
    **arches_params
)
```

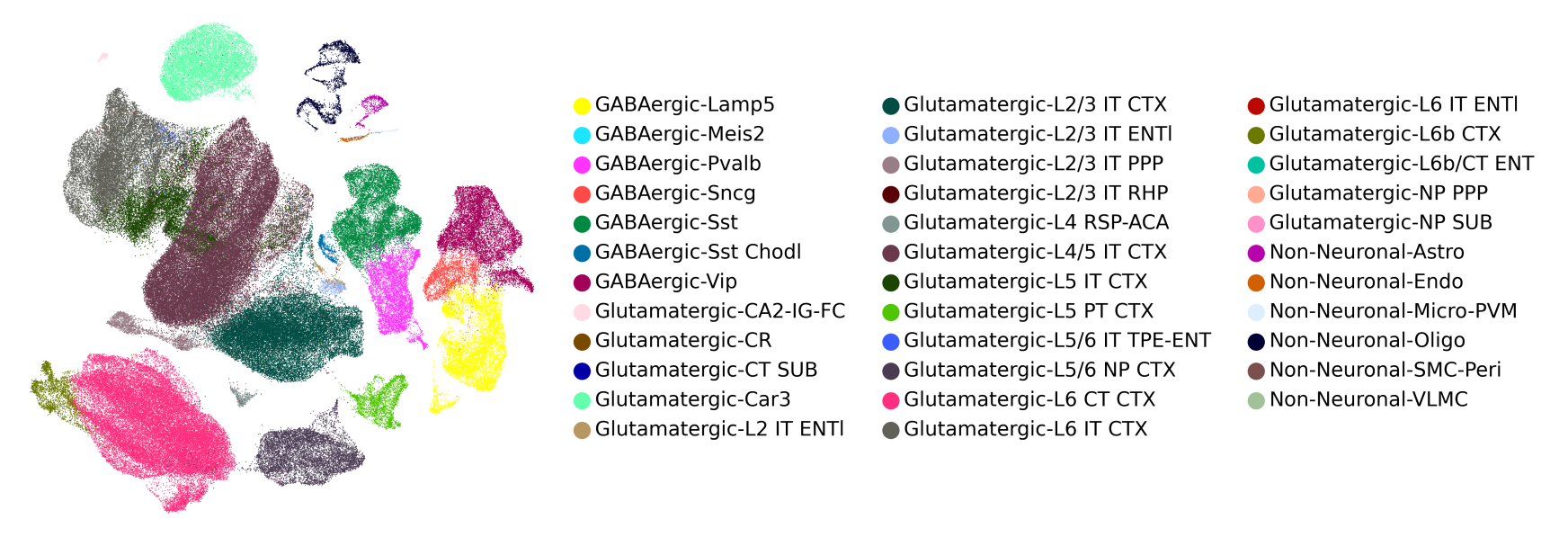
In [30]:
```python
vae_ref.train(max_epochs = 50, use_gpu=True)
```

```
GPU available: True, used: True
TPU available: False, using: 0 TPU cores
IPU available: False, using: 0 IPUs
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
Set SLURM handle signals.
Epoch 50/50: 100%|██████████| 50/50 [11:28<00:00, 13.77s/it, loss=3.46e+03, v_num=1]
```

In [31]:
```python
adata_ref.obsm["X_scVI"] = vae_ref.get_latent_representation()
sc.pp.neighbors(adata_ref, use_rep="X_scVI")
sc.tl.louvain(adata_ref)
sc.tl.umap(adata_ref)
```

```
computing neighbors
    finished: added to `.uns['neighbors']`
    `.obsp['distances']`, distances for each pair of neighbors
    `.obsp['connectivities']`, weighted adjacency matrix (0:00:32)
running Louvain clustering
    using the "louvain" package of Traag (2017)
    finished: found 36 clusters and added
    'louvain', the cluster labels (adata.obs, categorical) (0:01:09)
computing UMAP
    finished: added
    'X_umap', UMAP coordinates (adata.obsm) (0:04:42)
```

In [32]:
```python
with mpl.rc_context({'figure.dpi': 300}):
    sc.pl.umap(
        adata_ref,
        color=["cell_type"],
        frameon=False,
        ncols=1,
        legend_fontsize='xx-small'
        )
```



2. Train reference with SCANVI

In [33]:
```python
adata_ref.obs["labels_scanvi"] = adata_ref.obs["cell_type"].values
```

In [34]:
```python
vae_ref_scan = scvi.model.SCANVI.from_scvi_model(
    vae_ref,
    unlabeled_category="Unknown",
    labels_key="labels_scanvi",
)
```

In [35]:
```python
vae_ref_scan.train(max_epochs=50, use_gpu=True)
```

```
INFO     Training for 50 epochs.
GPU available: True, used: True
TPU available: False, using: 0 TPU cores
IPU available: False, using: 0 IPUs
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
Set SLURM handle signals.
Epoch 50/50: 100%|██████████| 50/50 [27:16<00:00, 32.73s/it, loss=3.62e+03, v_num=1]
```

In [36]:
```python
adata_ref.obsm["X_scANVI"] = vae_ref_scan.get_latent_representation()
```

In [37]:
```python
sc.pp.neighbors(adata_ref, use_rep="X_scANVI")
sc.tl.louvain(adata_ref)
sc.tl.umap(adata_ref)
```

```
computing neighbors
    finished: added to `.uns['neighbors']`
    `.obsp['distances']`, distances for each pair of neighbors
    `.obsp['connectivities']`, weighted adjacency matrix (0:00:14)
running Louvain clustering
    using the "louvain" package of Traag (2017)
    finished: found 35 clusters and added
    'louvain', the cluster labels (adata.obs, categorical) (0:01:20)
computing UMAP
    finished: added
    'X_umap', UMAP coordinates (adata.obsm) (0:04:44)
```

In [38]:
```python
scvi.model.SCANVI.from_scvi_model
```

Out[38]:
```
<bound method SCANVI.from_scvi_model of <class 'scvi.model._scanvi.SCANVI'>>
```

In [39]:
```python
with mpl.rc_context({'figure.dpi': 300}):
    sc.pl.umap(
        adata_ref,
        color=["cell_type"],
        frameon=False,
        ncols=1,
        legend_fontsize='xx-small'
    )
```

3. Transfer labels to query dataset

In [40]:
```python
adata_query.obs['labels_scanvi'] = vae_ref_scan.unlabeled_category_ # must add the unknown labels
```

In [41]:
```python
vae_q = scvi.model.SCANVI.load_query_data(
    adata_query,
    vae_ref_scan,
)
```

In [42]:
```python
vae_q.train(
    max_epochs=50,
    plan_kwargs=dict(weight_decay=0.0),
    check_val_every_n_epoch=10,
    use_gpu=True
)
```

```
INFO     Training for 50 epochs.
GPU available: True, used: True
TPU available: False, using: 0 TPU cores
IPU available: False, using: 0 IPUs
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
Set SLURM handle signals.
Epoch 50/50: 100%|██████████| 50/50 [07:08<00:00,  8.58s/it, loss=3.44e+03, v_num=1]
```

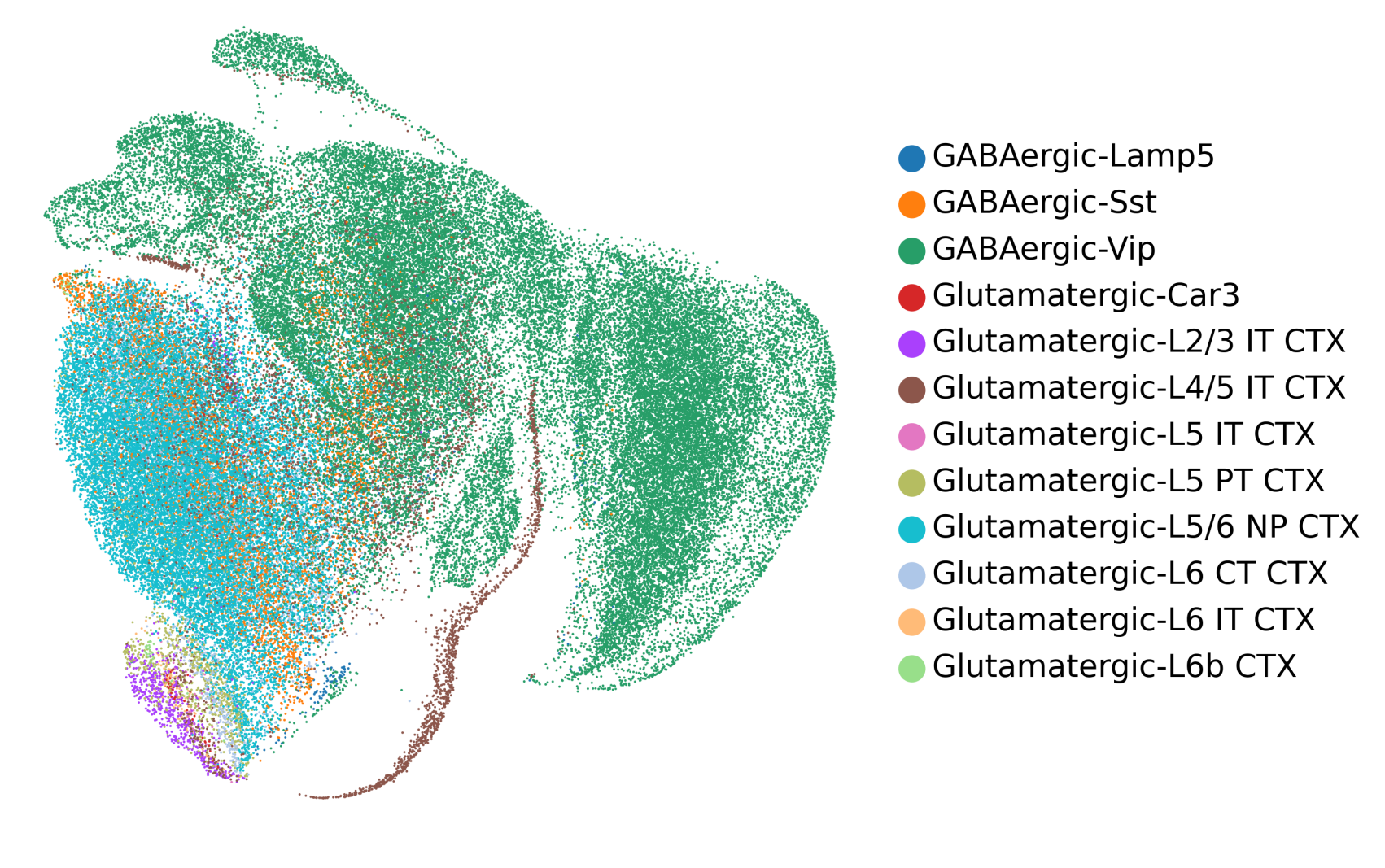
In [43]:
```python
adata_query.obsm["X_scANVI"] = vae_q.get_latent_representation()
adata_query.obs["predictions"] = vae_q.predict()
```

In [44]:
```python
sc.pp.neighbors(adata_query, use_rep="X_scANVI")
sc.tl.louvain(adata_query, key_added="louvain_scanvi")
sc.tl.umap(adata_query)
```

```
computing neighbors
    finished: added to `.uns['neighbors']`
    `.obsp['distances']`, distances for each pair of neighbors
    `.obsp['connectivities']`, weighted adjacency matrix (0:00:05)
running Louvain clustering
    using the "louvain" package of Traag (2017)
    finished: found 24 clusters and added
    'louvain_scanvi', the cluster labels (adata.obs, categorical) (0:00:22)
computing UMAP
    finished: added
    'X_umap', UMAP coordinates (adata.obsm) (0:01:29)
```

In [45]:
```python
with mpl.rc_context({'figure.dpi': 300}):
    sc.pl.umap(
        adata_query,
        color=["predictions"],
        frameon=False,
        ncols=1,
        legend_fontsize='xx-small'
        )
```

```
... storing 'labels_scanvi' as categorical
... storing 'predictions' as categorical
```



In [ ]: