

UM Cryo-ET Workshop

Wednesday 6/12: TomoDRGN

Overview

TomoDRGN is a deep learning tool for analyzing structural heterogeneity among particles imaged by cryogenic electron tomography (cryo-ET). TomoDRGN was developed as an extension from cryoDRGN, which has been widely applied to characterize structural heterogeneity in cryogenic electron microscopy (cryo-EM) datasets.

The core functionality tomoDRGN is comprised of two parts: (1) training a variational autoencoder neural network to learn structural heterogeneity among a dataset of particles that have been previously aligned by sub-tomogram averaging (STA), and (2) analyzing that model to identify structural heterogeneity trends and groups, quantifiable down to a per-particle level. The tomoDRGN network can be analyzed in both “latent space” (a low dimensional vector that summarizes the unique structural state of each particle) and in “volume space” (the real space 3-D volume unique to each particle). Useful insights may be gleaned from manually examining this structural heterogeneity directly within tomoDRGN, systematically quantifying heterogeneous states among the ensemble of volumes with tools such as MAVEn and SIREn, separating structurally distinct classes of particles for subsequent STA refinement, and correlating structurally heterogeneous states with spatial localization and geometry in the source tomograms.

This tutorial guides users through use of tomoDRGN in analyzing a widely used benchmark dataset: ribosomes imaged in sub-lethally chloramphenicol treated *Mycoplasma pneumoniae* cells (EMPIAR-10499). The tutorial illustrates the types of analyses performed in tomoDRGN’s primary methods manuscript¹. All inputs and outputs to this tutorial are available from EMPIAR-10499, EMPIAR-11843, and Zenodo, while the tomoDRGN software itself is available from GitHub.

References:

1. Powell, B.M., Davis, J.H. Learning structural heterogeneity from cryo-electron sub-tomograms with tomoDRGN. *Nature Methods* (2024). <https://doi.org/10.1038/s41592-024-02210-z>
2. Powell, B.M.; Brant, T.S.; Davis, J.H.; Mosalaganti, S. Rapid structural analysis of bacterial ribosomes in situ. *bioRxiv* (2024). <https://doi.org/10.1101/2024.03.22.586148>

Table of Contents

Overview	1
Table of Contents	2
1. Initialize the computational environment	3
2. Obtain raw data.....	4
3. Validate particle extraction	8
4. Train a full tomoDRGN network.....	10
5. Analyze the trained model to identify non-ribosomal particles.....	11
6. Alternative visualizations and particle identification strategies	17
7. Train a tomoDRGN network on a particle subset	19
8. Identify ribosomal structural heterogeneity.....	20
9. Validate interesting state particles.....	23
10. Iterative identification and refinement of species in RELION / M	25
11. Train a tomoDRGN network on a particle subset at intermolecular scale.....	27
12. Identify intermolecular structural heterogeneity	28
13. Validate membrane-associated ribosomes.....	31
14. Map tomoDRGN-generated intermolecular volumes to tomogram spatial context	34
15. Map tomoDRGN-generated intramolecular volumes to tomogram spatial context	37
What might come next?.....	39
Frequently asked questions	40
Supplemental: voxel_pca_umap.py	43

1. Initialize the computational environment

This tutorial requires the following software:

- tomoDRGN: <https://www.github.com/bpowell122/tomodrgn> (v0.2.2)
- RELION: <https://github.com/3dem/relion>
- ChimeraX: <https://www.rbvi.ucsf.edu/chimerax/>

You may find it useful to have a few terminals open while proceeding through this tutorial so as to avoid having to re-load each software at various points.

Terminal-1 (tomoDRGN)

```
$ mkdir /work/participant/tomodrgn_tutorial/  
$ cd /work/participant/tomodrgn_tutorial/  
$ source ~/conda_init.sh  
$ conda activate tomodrgn
```

Terminal-2 (tomoDRGN Jupyter notebook)

```
$ cd /work/participant/tomodrgn_tutorial/  
$ source ~/conda_init.sh  
$ conda activate tomodrgn  
$ jupyter notebook
```

Note: this may launch a web page that asks for a token that does not exist. If so, you can create a password first, then re-launch jupyter:

```
$ jupyter notebook password
```

Enter any password you choose (e.g. `cryoet`), then enter that password a second time to confirm.

```
$ jupyter notebook
```

Now the Jupyter notebook browser page should ask for a password instead of a token. Enter the password you just set.

Terminal-3 (RELION)

```
$ module load relion  
$ cd /work/participant/tomodrgn_tutorial/
```

Terminal-4 (ChimeraX)

```
$ chimeraX
```

Note that the DCV Viewer application may rebind copy and paste. On my Mac, copy/paste is rebound to [command]+[shift]+[C/V]

2. Obtain raw data

Several input data types from upstream processing are required to take full advantage of tomoDRGN's heterogeneity analysis, validation, and iterative processing potential. However, only some files are required for minimal functionality.

Strictly required

1. "Image series" / "particle series" subtomograms
 - Real-space 2-D projection images of each particle used by tomoDRGN
 - Warp and M offer this option via "Subtomogram Extraction > Image series"
2. Star file metadata for "image series" subtomograms
 - RELION v3 star file (i.e. no data_optics block due to Warp/M compatibility)
 - Each tilt image is referenced by one row. Tilt images of the same particle are identified by the rlnGroupName column

Strongly recommended






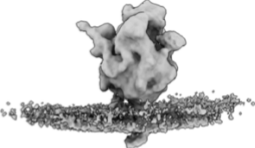
3. "Volume series" subtomograms
 - Real-space 3-D subtomograms of each particle used in RELION to validate structurally distinct particle sets
 - Warp and M default to this via "Subtomogram Extraction > Volume series"
4. Star file metadata for "volume series" subtomograms
 - RELION v3 star file (i.e. no data_optics block due to Warp/M compatibility)
 - Each particle is referenced by one row.

From EMPIAR and Zenodo

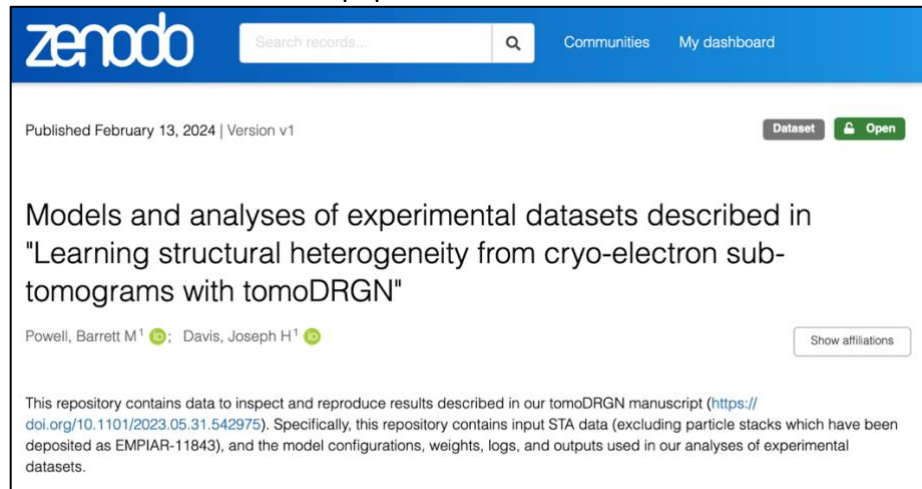
Both particle series and volume series subtomograms are pre-extracted and are publicly available for EMPIAR-10499 as described in the tomoDRGN methods paper:

EMPIAR-11843

Cryo electron tomography image- and volume-series subtomograms of chloramphenicol-treated mycoplasma pneumoniae

Entry authors:	Powell B.M.  , Davis J.H. 	Contains:  picked particles subtomograms
Publication:	Learning structural heterogeneity from cryo-electron subtomograms with tomoDRGN Powell BM  , Davis JH  <i>bioRxiv</i> (2023) DOI: 10.1101/2023.05.31.542975	
Experiment type:	EMDB	
Related EMD entry:	EMD-43287 (26.0Å)	
Related EMPIAR entry:	EMPIAR-10499	
Deposited:	2023-05-18	
Released:	2024-03-18	
Last modified:	2024-03-18	
Dataset size:	788.5 GB	
Dataset DOI:	10.6019/EMPIAR-11843	
Funding:	1. National Science Foundation (NSF, United States) 2046778 United States	
Experimental metadata:	Show more Download xml	

Pre-trained tomoDRGN models trained on these datasets are publicly available on Zenodo as described in the tomoDRGN methods paper:



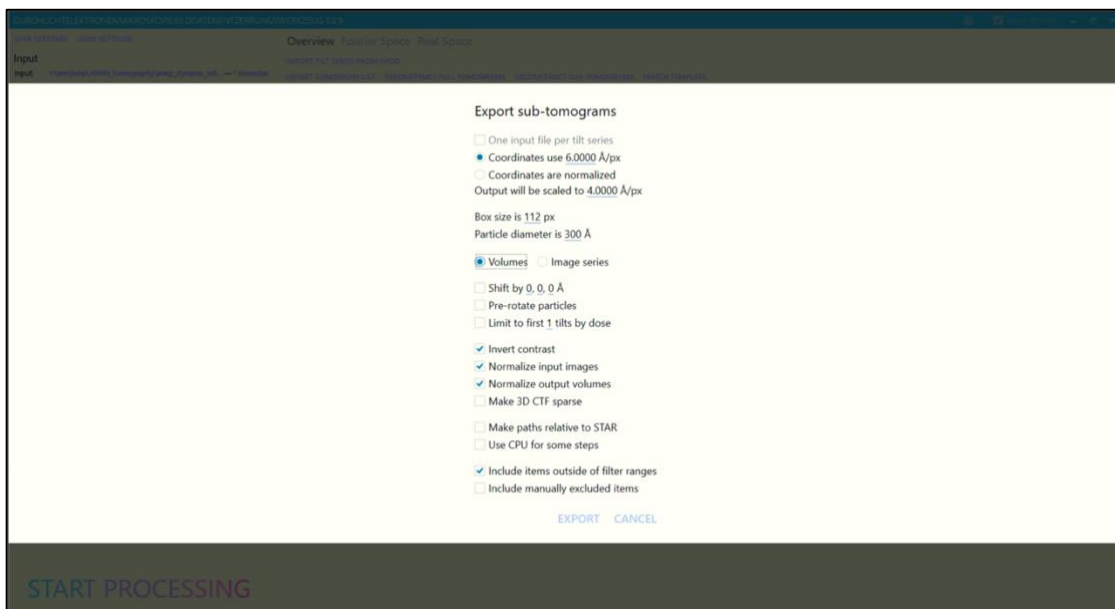
This tutorial will use a subset of the deposited datasets and models:

- <https://www.ebi.ac.uk/empiar/EMPIAR-11843/>
 - 11843/data/particleseries_box96_angpix3.7 (31 GB)
 - 11843/data/subtomo_box64_angpix6 (33 GB)
 - 11843/data/particleseries_box200_angpix3.7 (136 GB)
 - 11843/data/subtomo_box192_angpix4_train30_kmeans100_secdfribos_380ptcls (15 GB)
 - 11843/data/starfiles (883 MB)
- <https://zenodo.org/records/10093310>
 - files/04_dataset_empiar_10499.zip (17 GB)

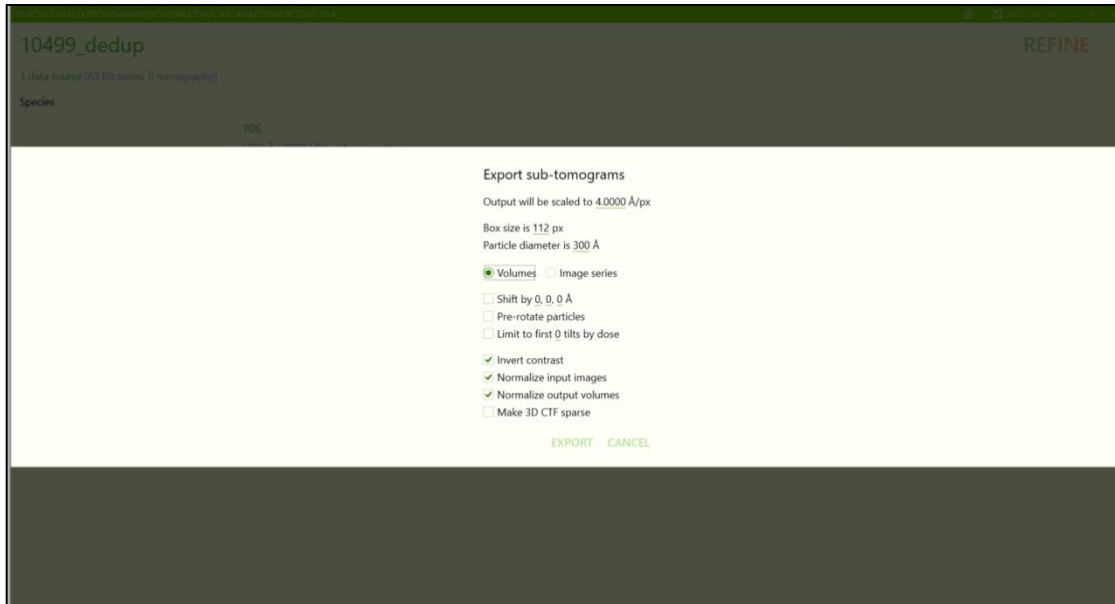
From particle extraction in Warp or M

Image series and volume series subtomograms can be extracted for your own datasets from Warp or M as follows:

Warp: In tomostar mode, select “export sub-tomograms”. Select the star file specifying particle locations to extract. The important parameter to set for tomoDRGN’s inputs is to choose either “volumes” or “image series”. The first option produces “volume series” subtomograms (3-D subtomogram volumes and CTF volumes suitable for refinement in RELION v3) or “image series” subtomograms (2-D projection images of each particle suitable for training tomoDRGN models), respectively. We recommend extracting both image series and volume series of the same particles such that you can do both tomoDRGN model training and subsequent particle subset validations and refinements. All other extraction parameters should be set following the Warp User’s Guide http://www.warpem.com/warp/?page_id=169.



M: Right click on the particle count of the species to export, and select “export particles”. Choose either “volumes” or “image series” for particle export as “volume series” or “image series”, respectively. Other parameters should be set following the Warp Users’s Guide http://www.warpem.com/warp/?page_id=169.



3. Validate particle extraction

Purpose

Once you have extracted (or downloaded a pre-extracted) subtomogram particleseries, it's a good idea to validate that the extraction worked correctly. Specifically, we're checking that the particle coordinates and poses used for extraction align with those used previously for STA. A quick way to confirm successful particle extraction is to generate a homogeneous reconstruction of the extracted particles, either via `tomodrgn backproject_voxel` or `tomodrgn train_nn`.

Performing a homogeneous reconstruction via `backproject_voxel`

TomoDRGN's `backproject_voxel` uses the specified particles to output an unfiltered consensus volume.

Terminal

```
$ mkdir 01_backproject
```

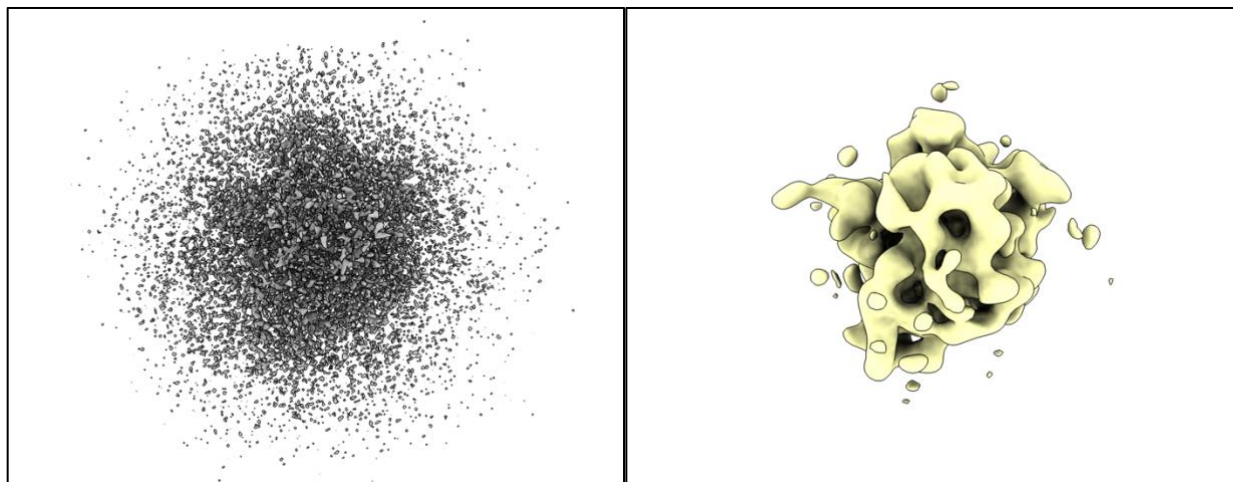
```
$ tomodrgn backproject_voxel \  
  /data/EMPIAR-11843/data/starfiles/10499_22k_box96_angpix3.7.star \  
  --datadir /data/EMPIAR-11843/data/particleseries_box96_angpix3.7 \  
  --first 5000 \  
  -o 01_backproject/backproject_first5000.mrc
```

Timing: 1 minute

```
$ relion_image_handler --i 01_backproject/backproject_first5000.mrc \  
  --o 01_backproject/backproject_first5000_lowpass30.mrc \  
  --lowpass 30
```

Chimerax:

Open `backproject_first5000_lowpass30.mrc`



The unfiltered backprojection of the first 5000 particle images is very noisy. After applying a strong lowpass filter, we clearly see ribosomal features. We can also notice that the data has the

correct dark/light convention (which we could (un)invert with `--uninvert-data`). TomoDRGN's `backproject` could be made less noisy by using more images (increasing the value passed to `--first`) or targeted to specific particles with `--ind`.

Performing a homogeneous reconstruction via `train_nn`

TomoDRGN's `train_nn` trains a decoder-only neural network using the specified particles to output a homogeneous consensus volume (i.e., no latent space or structural heterogeneity is learned). This is generally a legacy tool, and we recommend validating particle extraction with `backproject_voxel` instead. However, this command will introduce you to some of the arguments and outputs that will be produced for the full heterogeneous `tomodrgn` model training.

Terminal

```
$ tomodrgn train_nn \  
  /data/EMPIAR-11843/data/starfiles/10499_22k_box96_angpix3.7.star \  
  --datadir /data/EMPIAR-11843/data/particleseries_box96_angpix3.7/ \  
  --outdir 02_train_nn \  
  --dim 512 \  
  --layers 3 \  
  -n 50 \  
  --l-dose-mask
```

Timing: 10 mins to load, 10 mins per epoch, can kill partway through via `[control]+[c]`

```
[$ ls -l 02_train_nn/  
config.pkl  
reconstruct.0.mrc  
run.log  
weighting_scheme_0.png  
weights.0.pkl
```

4. Train a full tomoDRGN network

Purpose

Learning structural heterogeneity from a dataset is the core functionality of tomoDRGN. This requires training a neural network from scratch for each dataset to learn the structural heterogeneity features unique to a particular set of particles' images. Additional / iterative rounds of model training on progressively-smaller particle subsets are possible for deeper analyses.

Terminal

```
$ tomodrnn train_vae \  
  /data/EMPIAR-11843/data/starfiles/10499_22k_box96_angpix3.7.star \  
  --datadir /data/EMPIAR-11843/data/particleseries_box96_angpix3.7/ \  
  --outdir 03_train_vae \  
  --enc-dim-A 256 \  
  --enc-layers-A 3 \  
  --out-dim-A 128 \  
  --enc-dim-B 256 \  
  --enc-layers-B 3 \  
  --zdim 128 \  
  --dec-dim 256 \  
  --dec-layers 3 \  
  -n 50 \  
  --l-dose-mask \  
  --recon-dose-weight \  
  --recon-tilt-weight
```

Timing: 5 mins to load, 10 mins per epoch, kill partway through

```
ls -l 03_train_vae/  
config.pkl  
run.log  
weighting_scheme_0.png  
weights.0.pkl  
z.0.pkl
```

5. Analyze the trained model to identify non-ribosomal particles

Purpose

The input dataset very likely contains non-ribosomal particles (e.g. other macromolecules, membranes, ice, metallic sputter, random noise). TomoDRGN's expressive model can aid in robustly detecting and removing these particles which otherwise consume the model's representation capacity of true ribosomal structural heterogeneity. We can separate "good" from "non-ribosomal" particles by a variety of methods in latent or real space. We describe and illustrate several of these methods here, as some may work better or worse on your particular dataset.

Standard tomoDRGN model analysis

We first copy the model weights and resulting particle latent embeddings from the 49th epoch of training that were precalculated for you.

Terminal:

```
$ cp /data/04_dataset_empiar_10499/01_tomodrgn/01_training_and_analysis/27_vae_box96_256x3_128_256x3_128_256x3_b1_gaussian/weights.49.pkl 03_train_vae/
```

```
$ cp /data/04_dataset_empiar_10499/01_tomodrgn/01_training_and_analysis/27_vae_box96_256x3_128_256x3_128_256x3_b1_gaussian/z.49.pkl 03_train_vae/
```

```
$ tomodrgn analyze \  
    03_train_vae \  
    49 \  
    --Apix 3.7 \  
    --flip \  
    --ksample 100
```

Timing: 1 minute

```
$ ls -l 03_train_vae/analyze.49/  
kmeans100  
pc1  
pc2  
tomoDRGN_viz+filt.ipynb  
umap.pkl  
umap.png  
umap_hexbin.png  
z_pca.png  
z_pca_hexbin.png
```

The `tomodrgn analyze` command evaluates a particular model (here, `03_train_vae`) at a particular epoch of training (here, epoch 49). The latent embeddings of all particles are subjected to principal component analysis (`z_pca.png`) and UMAP (`umap.png`, `umap.pkl`) dimensionality reduction. Volumes are generated from latent embeddings that sample deciles along each of the first two latent space principal component axes (`pc1/`, `pc2/`). The latent space is more

broadly sampled by k-means clustering, and the centroid latent embedding of each k-means class is used to generate “k100 centroid volumes”.

The k100 centroid volumes are useful to visually inspect the types of structural heterogeneity learned from the dataset. Increasing the number of k-means clusters will give more granular insights to per-particle structural heterogeneity at the trade-off of requiring more manual inspection to characterize. Due to the random initialization of the k-means sampling, we will use pre-generated k-means classes from Zenodo to be consistent in further analysis.

```
$ cp -R  
/data/04_dataset_empiar_10499/01_tomodrgn/01_training_and_analysis/27_vae_box96_256x3_128_256x3_128_256x3_b1_gaussian/analyze.49 03_train_vae/analyze.49.published
```

```
$ cp 03_train_vae/analyze.49/tomoDRGN_viz+filt.ipynb  
03_train_vae/analyze.49.published/tomoDRGN_viz+filt.ipynb
```

Identify non-ribosomal particles by visual inspection of k100 volumes

ChimeraX

```
open "03_train_vae/analyze.49.published/kmeans100/*.mrc"
```

```
volume all level 0.018
```

```
surface dust all size 6
```

```
mseries slider all
```

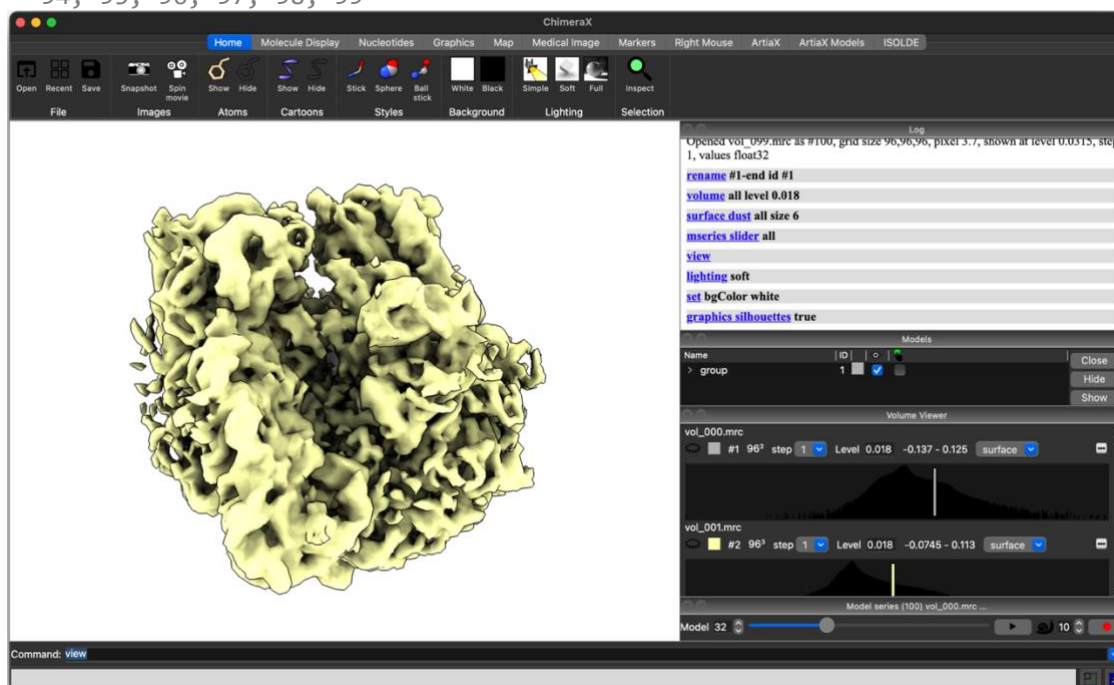
```
visually inspect for 70S / 50S / non-ribosomal
```

```
my k100 labels for non-ribosomal:
```

```
0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 15, 16, 17, 18, 92, 93
```

```
my k100 labels for 50S:
```

```
94, 95, 96, 97, 98, 99
```



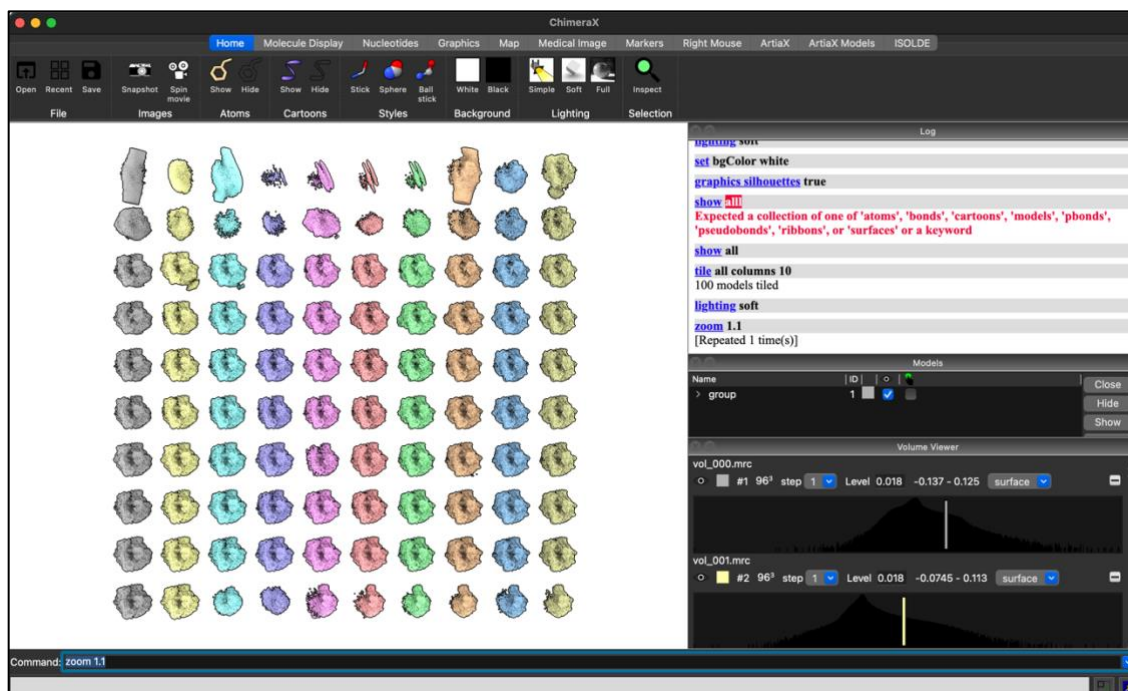
Flip through the mseries to see all 100 volumes. Note large and systematic changes of density. At this stage we're aiming to annotate "ribosomal" (which could be 50S or 70S) separate from "non-ribosomal". These junk particles are generally noisy, have less defined structure than good particles, and can have very large or small density at a reasonable isosurface for the good particles.

Keep in mind that tomoDRGN indexes volumes in a 0-based system, but the mseries is 1-indexed!

Sometimes it can be easier to see large scale systematic differences by tiling all volumes instead of flipping through an mseries.

show all

tile all columns 10



Generate indices of good particles based on k100 annotation

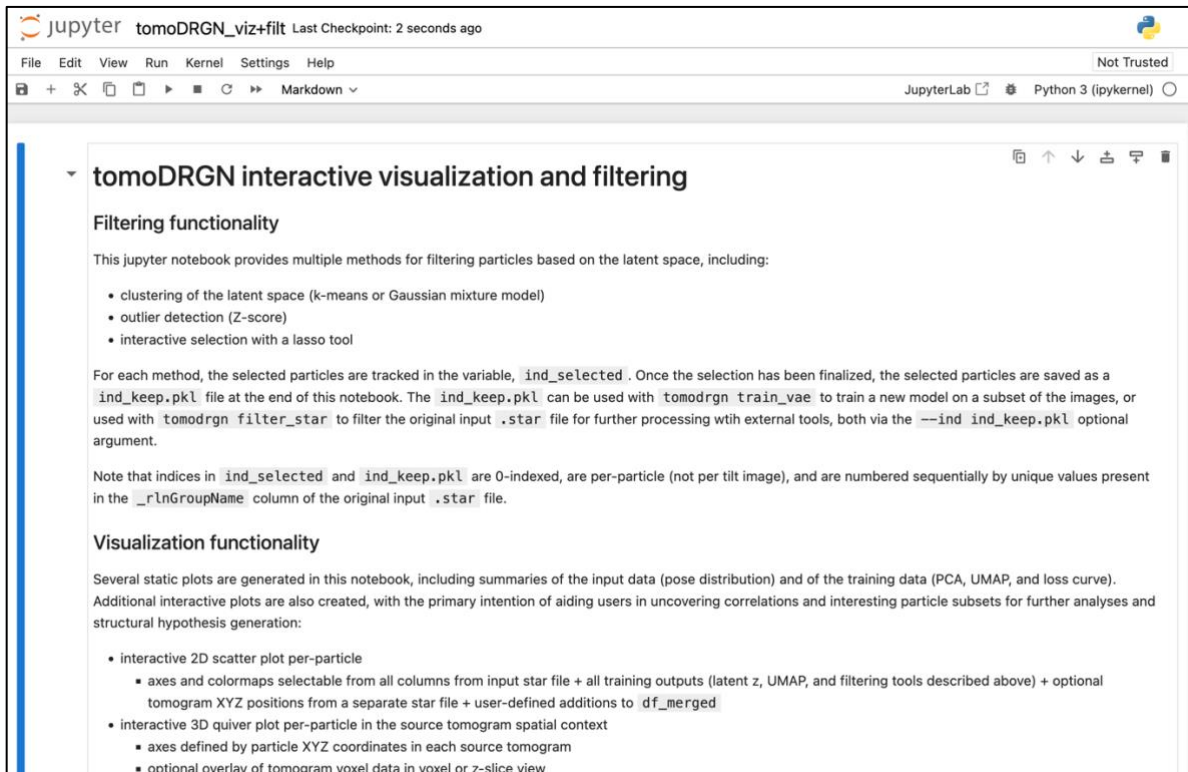
Good particle indices can be generated with the standard analysis Jupyter notebook (which offers more features, visualizations, and alternative filtering options), or directly at the command line via a few python commands (which can be more convenient for quick annotation).

Jupyter notebook

Open in jupyter 03_train_vae/analyze.49/tomoDRGN_viz+filt.ipynb

See this page for a useful reference of Jupyter notebook tools:

https://www.edureka.co/blog/wp-content/uploads/2018/10/Jupyter_Notebook_CheatSheet_Edureka.pdf



When reading in kmeans100, update path to:

```
kmeans_labels = utils.load_pkl(f'{WORKDIR}/analyze.{EPOCH}.published/kmeans{K}/labels.pkl')
```

```
kmeans_centers = np.loadtxt(f'{WORKDIR}/analyze.{EPOCH}.published/kmeans{K}/centers.txt')
```

```
# Load kmeans
K = 100 # or user defined if re-running kmeans
kmeans_labels = utils.load_pkl(f'{WORKDIR}/analyze.{EPOCH}.published/kmeans{K}/labels.pkl')
kmeans_centers = np.loadtxt(f'{WORKDIR}/analyze.{EPOCH}.published/kmeans{K}/centers.txt')
```

When reading in volumeseries star file, set path to:

```
path_to_volseries_star = '/data/EMPIAR-11843/data/starfiles/10499_22k_box64_angpix6_volumeseries.star'
```

```
# Optionally load a RELION3.0 volumeseries star file from Warp/M (to get particle positions within tomograms)
# Starfile must reference the same set of particles referenced by the starfile used for tomodrgn train_vae

### USER INPUT: metadata to find particle positions, merge with training df, and rescale to tomogram pixel size
path_to_volseries_star = '../..../data/starfiles/10499_22k_box64_angpix6_volumeseries.star' # absolute path to volume series star file
star_from_M = False # True if star file from M; False if star file from Warp subtomogram volumeseries
tomo_max_xyz_nm = (680, 680, 510) # dimensions in nm of reconstructed tomograms for later interactive visualization
tomo_pixelsize = 10 # pixel size of reconstructed tomogram in A/px
starfile_pixelsize = 6 # pixel size in volumeseries star file, CoordinateX,Y columns, in A/px
```

When selecting particles based on k-means clustering, set clusters to:

```
cluster_ids = [0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 15, 16, 17, 18, 92, 93]
```

Select particles based on k-means clustering

```
cluster_ids = [0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 15, 16, 17, 18, 92, 93] # USER INPUT: integer I.D. of clusters to select, 0 and
ind_selected = select_clusters(kmeans_labels, cluster_ids)
ind_selected_not = invert_selection(kmeans_labels, ind_selected)
```

Run the “select particles based on GMM cluster” and “select particles based on z-norm outliers” sections to try other ways to classify particles based on latent embeddings. When finished, re-run the “select particles based on k-means clustering” section to make sure those selections are carried forward. The important thing is to make sure that the cell in the screenshot above is run last before proceeding to “interactive visualization” section below, as this sets the `cluster_ids` variable correctly.

Run the “interactive visualization” first few cells to produce `df_merged`, then skip “interactive selection”, “view tilt images from selected particles”, and “View particle distributions in tomogram context” sections (do not have tomograms available)

Note that there is a bug with interactive selection in some versions of jupyter notebook and associated widgets that causes the notebook to stop producing output after the interactive plotting + selection cell has been run. To recover from this, restart the notebook (and return to the beginning of the Jupyter notebook section of this tutorial).

In “Save selection indices” section, update selection/not:

```
ind_keep = ind_selected_not
ind_bad = ind_selected
```

```
# Set selection as either the kept or bad particles (for file naming purposes)
ind_keep = ind_selected_not # or ind_selected_not
ind_bad = ind_selected # or ind_selected
```

Finish with saving `ind_keep` and `ind_bad` pkl files.

```
# Path to save index .pkl for selected particles
SAVE_PATH = f'{WORKDIR}/ind_keep.{len(ind_keep)}_particles.pkl'
utils.save_pkl(ind_keep, SAVE_PATH)
print(f'Wrote {os.path.abspath(SAVE_PATH)}')

Wrote /nbackup/users/bmp/experiments/UM_cryoET_workshop/outputs/03_train_vae/ind_keep.20981_particles.pkl

# Path to save index .pkl for non-selected particles
SAVE_PATH = f'{WORKDIR}/ind_bad.{len(ind_bad)}_particles.pkl'
utils.save_pkl(ind_keep, SAVE_PATH)
print(f'Wrote {os.path.abspath(SAVE_PATH)}')

Wrote /nbackup/users/bmp/experiments/UM_cryoET_workshop/outputs/03_train_vae/ind_bad.1310_particles.pkl
```

Don't close notebook yet!

Terminal:

```
$ python
>>> import numpy as np
>>> from tomodrjn import utils
>>> k100_labels =
utils.load_pkl('03_train_vae/analyze.49.published/kmeans100/labels.pkl')
>>> labels_nr = [0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 15, 16, 17, 18, 92,
93]
>>> ind_nr = [ind for ind, label in enumerate(k100_labels) if label in labels_nr]
```

```
>>> ind_nr = np.array(ind_nr)
>>> utils.save_pkl(ind_nr, '03_train_vae/ind_nr.1310_particles.pkl')
>>> ind_ribo = [ind for ind, label in enumerate(k100_labels) if label not in
labels_nr]
>>> ind_ribo = np.array(ind_ribo)
>>> utils.save_pkl(ind_ribo, '03_train_vae/ind_ribo.20981_particles.pkl')
>>> exit()
```


6. Alternative visualizations and particle identification strategies

Purpose

While particles of distinct structural states can often be well separated by the latent space k100 sampling described above, alternative approaches are possible and may work better for other datasets. These include latent space gaussian mixture model (GMM) clustering, latent space outlier identification, and volume space voxel PCA and UMAP.

Latent space analysis

The latent space can be analyzed and grouped by any number of clustering approaches. The Jupyter notebook includes support for latent space GMM clustering, and other clustering approaches can be used. The magnitude of each particle's latent encoding can also be correlated with "junk" particle identity, whereby junk particles can have latent embeddings on the periphery of the main distribution centered near zero.

Jupyter notebook

Try different clustering settings and random seeds in the "GMM clustering" section. Try different `zscore` latent magnitude cutoffs in the "Filter by latent outliers" section.

Volume space analysis

While the latent space learned by tomoDRGN is a low dimensional representation of the structural heterogeneity captured by each dataset, sometimes directly analyzing the ensemble of tomoDRGN-generated heterogeneous volumes can give more direct and informative insights. Here we analyze a volume ensemble of all 22,291 unique particles (at box size 32px) by PCA (to the first 128 components) followed by UMAP dimensionality reduction. A volume ensemble generated in this way can also be used as inputs to MAVEn or SIREn.

Terminal

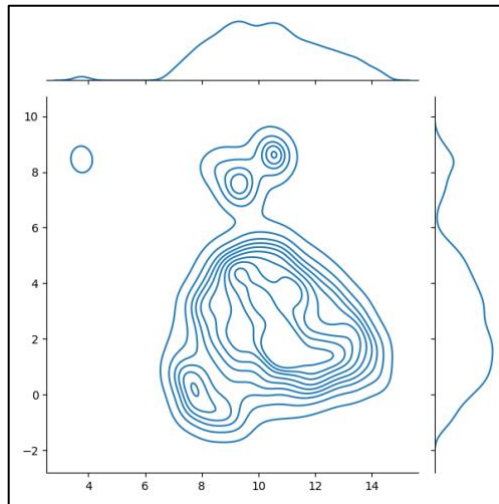
```
$ mkdir 03_train_vae/analyze.49.published/all_vols_box32
```

```
$ tomodrgn eval_vol -w 03_train_vae/weights.49.pkl \  
  -c 03_train_vae/config.pkl \  
  -o 03_train_vae/analyze.49.published/all_vols_box32 \  
  --zfile 03_train_vae/z.49.pkl \  
  --downsample 32 \  
  --Apix 11.1 \  
  --flip \  
  -b 64
```

Timing: ~3 minutes

```
$ python /sw/tomodrgn/0.2.2/utis/voxel_pca_umap.py \  
  --vol-dir 03_train_vae/analyze.49.published/all_vols_box32 \  
  --out-dir 03_train_vae/analyze.49.published/all_vols_box32_analysis \  
  --num-pcs 128
```

Timing: ~5 mins



The per-particle volume space voxel-pca-umap embeddings can also be loaded into the Jupyter notebook for interactive analysis or to use as the basis for clustering (rather than the latent space embeddings).

Create a new cell in the Jupyter notebook at the better of the “Interactive visualization” section. Insert and run the following code:

```
voxel_pca_umap =  
utils.load_pkl('03_train_vae/analyze.49.published/all_vols_box32_analysis/voxel_pc_umap.pkl')  
df_merged['voxel_pca_umap1'] = voxel_pca_umap[:,0]  
df_merged['voxel_pca_umap2'] = voxel_pca_umap[:,1]
```

You could at this point create many types of plots, clustering, etc by cross referencing the many metadata details available for each particle in the `df_merged` dataframe.

Now can close notebook :)

7. Train a tomoDRGN network on a particle subset

Purpose

Now that the non-ribosomal particles have been identified, we can train a new tomoDRGN model on a more purely-ribosomal particle stack to more fully leverage the learning capacity of the network.

Terminal

```
$ tomodrnn train_vae \  
  /data/EMPIAR-11843/data/starfiles/10499_22k_box96_angpix3.7.star \  
  --datadir /data/EMPIAR-11843/data/particleseries_box96_angpix3.7/ \  
  --ind 03_train_vae/ind_keep.20981_particles.pkl \  
  --outdir 04_train_vae_filtered \  
  --enc-dim-A 256 \  
  --enc-layers-A 3 \  
  --out-dim-A 128 \  
  --enc-dim-B 256 \  
  --enc-layers-B 3 \  
  --zdim 128 \  
  --dec-dim 256 \  
  --dec-layers 3 \  
  -n 50 \  
  --l-dose-mask \  
  --recon-dose-weight \  
  --recon-tilt-weight
```

Timing: 5 mins to load, 10 mins per epoch, kill partway through

8. Identify ribosomal structural heterogeneity

Purpose

Equipped with a tomoDRGN model trained on a purely-ribosomal dataset, we can now analyze that model for structural heterogeneity. This analysis will use many of the same commands and tools introduced earlier in the non-ribosomal particle filtration section

Standard tomoDRGN model analysis

Terminal:

```
$ cp /data/04_dataset_empiar_10499/01_tomodrgn/01_training_and_analysis/28_vae_box96_256x3_128_256x3_128_256x3_b1_gaussian_ind20981/weights.49.pkl 04_train_vae_filtered/
```

```
$ cp /data/04_dataset_empiar_10499/01_tomodrgn/01_training_and_analysis/28_vae_box96_256x3_128_256x3_128_256x3_b1_gaussian_ind20981/z.49.pkl 04_train_vae_filtered/
```

```
$ tomodrgn analyze \  
    04_train_vae_filtered \  
    49 \  
    --Apix 3.7 \  
    --flip \  
    --ksample 100
```

Timing: 1 minute

As described earlier for the non-ribosomal particle analysis, due to the randomness of the k-means clustering during tomodrgn analyze, we will examine k100 volumes from a precomputed analysis for consistent analysis.

```
$ cp -R /data/04_dataset_empiar_10499/01_tomodrgn/01_training_and_analysis/28_vae_box96_256x3_128_256x3_128_256x3_b1_gaussian_ind20981/analyze.49 04_train_vae_filtered/analyze.49.published
```

```
$ cp 04_train_vae_filtered/analyze.49/tomoDRGN_viz+filt.ipynb 04_train_vae_filtered/analyze.49.published/tomoDRGN_viz+filt.ipynb
```

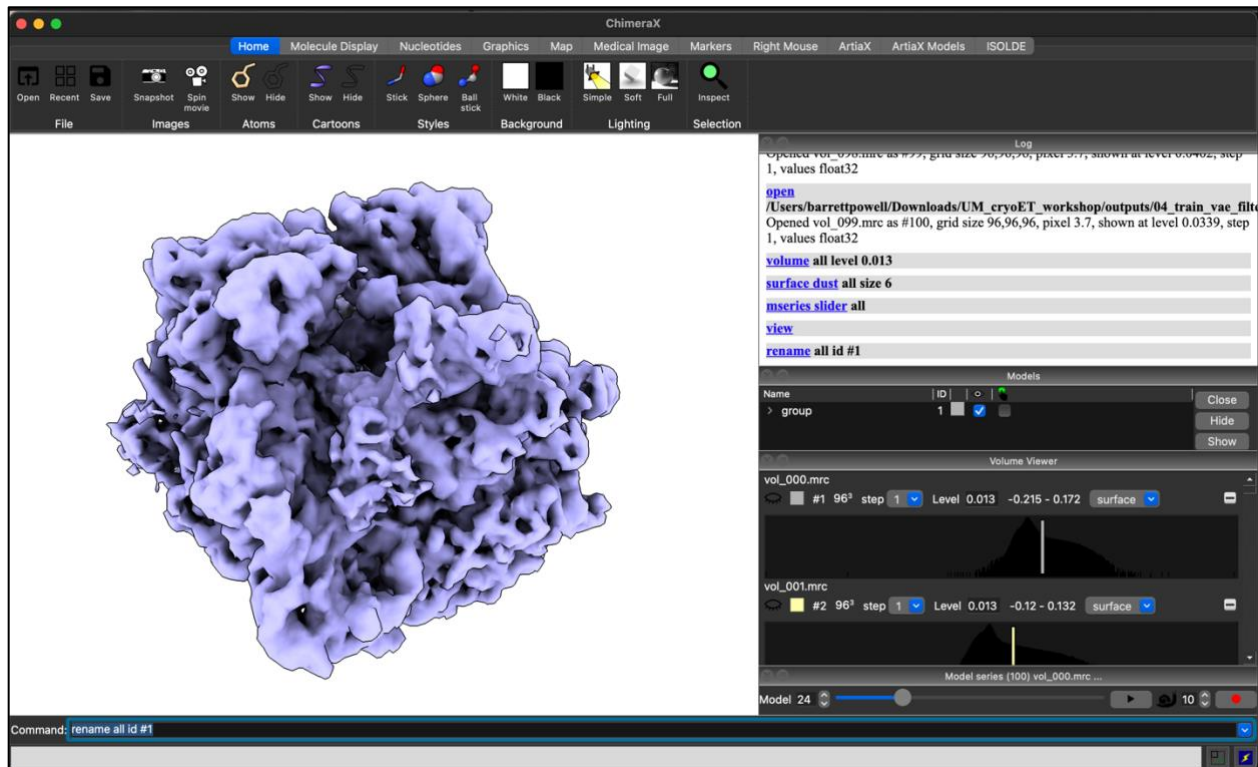
Identify biologically interesting states by visual inspection of k100 volumes

At this stage of analysis, we expect to see extensive ribosomal structural heterogeneity. This will principally involve heterogeneity related to translational states (A- and P- site tRNAs, EF-Tu bound T-site tRNA)

ChimeraX

```
open "04_train_vae_filtered/analyze.49.published/kmeans100/*.mrc"  
volume all level 0.013  
surface dust all size 6  
mseries slider all  
visually inspect for biologically interesting states
```

50S: vol_005.mrc vs vol_021.mrc
 SSU rotation with H17 motion: vol_021.mrc vs vol_023.mrc
 P-only: vol_031.mrc or vol_093.mrc
 EFTu-P: vol_023.mrc
 AP: vol_012.mrc
 peripheral density: vol_088.mrc
 visually inspect for 50S or EF-Tu k100 classes:
 my k100 labels for 50S
 2, 3, 4, 5, 7, 9
 My k100 labels for EF-Tu
 22, 23, 24, 25, 26, 27, 28, 29, 33, 61, 62, 75, 88



It can be very useful to have an atomic model to guide interpretation of heterogeneous volumes (where do you observe extra density unaccounted for by the model, where is the model fitting the density well, etc.). This dataset was originally used to solve PDBs: 7ph9, 7pha, 7phb, 7phc. You can easily open atomic models in ChimeraX with:

open 7phb from pdb

Generate 50S particle indices from k100 annotation

As above when identifying good vs non-ribosomal particle indices, we illustrate both a more flexible and comprehensive Jupyter notebook based method for generating indices, and a quick interactive python session method as well.

Jupyter notebook

Open in jupyter

04_train_vae_filtered/analyze.49/tomoDRGN_viz+filt.ipynb

When reading in kmeans100, update path to:

```
kmeans_labels = utils.load_pkl(f'{WORKDIR}/analyze.{EPOCH}.published/kmeans{K}/labels.pkl')
kmeans_centers = np.loadtxt(f'{WORKDIR}/analyze.{EPOCH}.published/kmeans{K}/centers.txt')
```

When reading in volumeseries star file, set path to:

```
path_to_volseries_star = '/data/EMPIAR-11843/data/starfiles/10499_22k_box64_angpix6_volumeseries.star'
```

When selecting particles based on k-means clustering, set clusters to:

```
cluster_ids = [2, 3, 4, 5, 7, 9]
```

Run the “select particles based on GMM cluster” and “select particles based on z-norm outliers” sections to try other ways to classify particles based on latent embeddings. When finished, re-run the “select particles based on k-means clustering” section to make sure those selections are carried forward. The important thing is to make sure that the cell in the screenshot above is run last before proceeding to “interactive visualization” section below, as this sets the `cluster_ids` variable correctly.

Run the “interactive visualization” first few cells to produce `df_merged`, then skip “interactive selection”, “view tilt images from selected particles”, and “View particle distributions in tomogram context” sections (do not have tomograms available)

Update the path naming the indices to save:

```
SAVE_PATH = f'{WORKDIR}/ind_keep.{len(ind_keep)}_particles_50S.pkl'
```

Terminal:

```
$ python
>>> import numpy as np
>>> from tomodrgrn import utils
>>> k100_labels =
utils.load_pkl('04_train_vae_filtered/analyze.49.published/kmeans100/labels.pkl')
>>> labels_50S = [2, 3, 4, 5, 7, 9]
>>> ind_50S = [ind for ind, label in enumerate(k100_labels) if label in labels_50S]
>>> ind_50S = np.array(ind_50S)
>>> ind_reindexing = utils.load_pkl('03_train_vae/ind_keep.20981_particles.pkl')
>>> ind_50S_reindexed = ind_reindexing[ind_50S]
>>> utils.save_pkl(ind_50S, '04_train_vae_filtered/ind_50S.615_particles.pkl')
>>> exit()
```

9. Validate interesting state particles

Purpose

While we have yet to observe a “structural hallucination” produced by tomoDRGN across several datasets, it is a reassuring good practice to validate observed structural heterogeneity of a subset of particles by performing a more traditional reconstruction of that particle subset to reproduce that particular structural state. Here we illustrate this process for the 50S ribosomal subunit population of the dataset.

Filter volume series star file by 50S particle indices

As the volume series star file can be used for downstream analysis in RELION, it is convenient to filter it to the particles of the structural state in question (here, 50S ribosomes).

Terminal

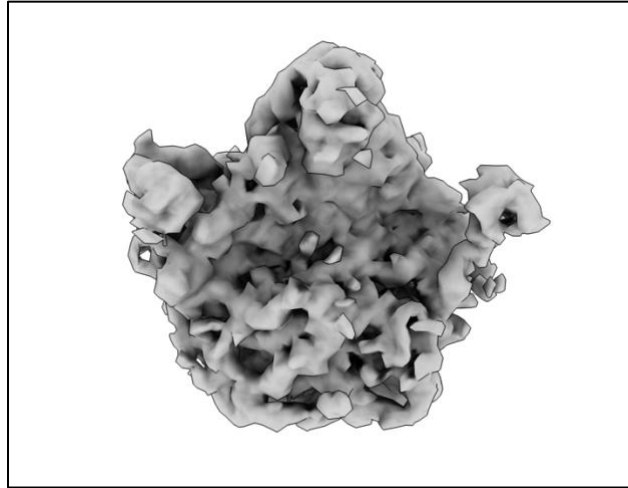
```
$ tomodrgn filter_star \  
  /data/EMPIAR-11843/data/starfiles/10499_22k_box64_angpix6_volumeseries.star \  
  --ind 04_train_vae_filtered/ind_keep.615_particles_50S.pkl \  
  --ptcl-id-col index \  
  -o 04_train_vae_filtered/10499_22k_box64_angpix6_volumeseries_615_particles_50S.star
```

Validate structural state of isolated particles

We next perform a homogeneous reconstruction of the specified 50S particles using RELION to validate their 50S structural state.

Terminal

```
$ mkdir -p 04_train_vae_filtered/validation/backproject_50S  
  
$ sed -i 's/./\subtomo_box64_angpix6/\data\EMPIAR-11843\data\subtomo_box64_angpix6/g'  
04_train_vae_filtered/10499_22k_box64_angpix6_volumeseries_615_particles_50S.star  
  
$ mpirun -n 9 relion_reconstruct_mpi \  
  --i  
04_train_vae_filtered/10499_22k_box64_angpix6_volumeseries_615_particles_50S.star \  
  --o  
04_train_vae_filtered/validation/backproject_50S/10499_22k_box64_angpix6_volumeseries_615_  
particles_50S_reconstruct.mrc \  
  --3d_rot \  
  --ctf \  
  --maxres 15  
Timing: 1 minute
```



10. Iterative identification and refinement of species in RELION / M

Purpose

Once a set of particles bearing distinct structural features have been identified (using tomoDRGN) and validated (using RELION or other traditional reconstruction approaches), one may want to improve particle STA parameters following this new structural reference model. This is typically done in RELION and/or M.

Isolate 50S particles

We already did this above when we filtered the volume series star file to just 50S particles!

Run RELION 3D auto refine

Because the 50S ribosome is so substantially different than the 70S against which it was refined by STA, it is worthwhile to re-refine the 50S particles against a 50S reference volume to produce optimal particle poses.

Terminal

```
$ mkdir 04_train_vae_filtered/validation/refine3d_50S

$ mpirun -n 9 --bind-to none relion_refine_mpi \
  --i
04_train_vae_filtered/10499_22k_box64_angpix6_volumeseries_615_particles_50S.star \
  --o 04_train_vae_filtered/validation/refine3d_50S/run
  --ref
04_train_vae_filtered/validation/backproject_50S/10499_22k_box64_angpix6_volumeseries_615_
particles_50S_reconstruct.mrc \
  --auto_refine \
  --split_random_halves \
  --firstiter_cc \
  --ini_high 60 \
  --dont_combine_weights_via_disc \
  --preread_images \
  --pool 3 \
  --pad 2 \
  --skip_gridding \
  --ctf \
  --particle_diameter 300 \
  --flatten_solvent \
  --zero_mask \
  --oversampling 1 \
  --healpix_order 2 \
  --auto_local_healpix_order 4 \
  --offset_range 5 \
  --offset_step 2 \
  --sym C1 \
  --low_resol_join_halves 40 \
  --norm \
  --scale \
```

```
--j 4 \  
--gpu "" \  
--preread_images
```

Timing: ~20 minutes (18A unmasked FSC final)

We could also import into M and refine as a distinct species (preferably alongside the 70S particles to constrain the refinement).

11. Train a tomoDRGN network on a particle subset at intermolecular scale

Purpose

The ribosomal heterogeneity model analyzed earlier by k100 volume inspection contained several volumes bearing features that appeared truncated by the limits of the reconstructed box. As these ribosomal particles were imaged *in situ* in a cell, such particle-adjacent density could plausibly derive from systematic intermolecular structural heterogeneity of the immediate structural neighborhood of each ribosome. To better characterize this intermolecular heterogeneity, we can re-extract our particles at a larger real space box size and train a new tomoDRGN model to learn and analyze intermolecular heterogeneity.

All 22,291 particles were pre-extracted at box size 200px and pixel size 3.7Å/px and are available from EMPIAR-11843. This means we need to apply the same particle filtering indices that we derived in the 03_train_vae analysis.

Terminal

```
$ tomodrpn train_vae \  
  /data/EMPIAR-11843/data/starfiles/10499_22k_box200_angpix3.7.star \  
  --datadir /data/EMPIAR-11843/data/particleseries_box200_angpix3.7 \  
  --ind 03_train_vae/ind_keep.20981_particles.pkl \  
  --outdir 05_train_vae_intermol_filtered \  
  --enc-dim-A 256 \  
  --enc-layers-A 3 \  
  --out-dim-A 128 \  
  --enc-dim-B 256 \  
  --enc-layers-B 3 \  
  --zdim 128 \  
  --dec-dim 256 \  
  --dec-layers 3 \  
  -n 50 \  
  --l-dose-mask \  
  --recon-dose-weight \  
  --recon-tilt-weight \  
  --lazy
```

Timing: ~5 mins to load, ~100 mins per epoch (due to lazy, due to 129 GiB RAM requirement), kill partway through

12. Identify intermolecular structural heterogeneity

Purpose

As is hopefully becoming familiar, once we have a trained tomoDRGN model, we will inspect a survey of its volumes to see what sorts of heterogeneity have been learned. Here, because we trained the model on a larger real space box containing each particle, we expect the primary modes of learned heterogeneity to be on an intermolecular scale instead of an intramolecular one (i.e., what types of densities systematically surround each ribosome, rather than what densities heterogeneously comprise each ribosome).

Standard tomoDRGN model analysis

Terminal

```
$ cp
/data/04_dataset_empiar_10499/01_tomodrgn/01_training_and_analysis/30_vae_box200_256x
3_128_256x3_128_256x3_b1_gaussian_ind20981_inter/weights.49.pkl
05_train_vae_intermol_filtered/
```

```
$ cp
/data/04_dataset_empiar_10499/01_tomodrgn/01_training_and_analysis/30_vae_box200_256x
3_128_256x3_128_256x3_b1_gaussian_ind20981_inter/z.49.pkl
05_train_vae_intermol_filtered/
```

```
$ tomodrgn analyze \
    05_train_vae_intermol_filtered \
    49 \
    --Apix 3.7 \
    --ksample 100 \
    --flip
```

Timing: ~4 minutes

```
$ cp -R
/data/04_dataset_empiar_10499/01_tomodrgn/01_training_and_analysis/30_vae_box200_256x
3_128_256x3_128_256x3_b1_gaussian_ind20981_inter/analyze.49
05_train_vae_intermol_filtered/analyze.49.published
```

```
$ cp 05_train_vae_intermol_filtered/analyze.49/tomoDRGN_viz+filt.ipynb
05_train_vae_intermol_filtered/analyze.49.published/tomoDRGN_viz+filt.ipynb
```

As described earlier, due to the randomness of the k-means clustering during tomodrgn analyze, we will examine k100 volumes from a precomputed analysis for consistent analysis.

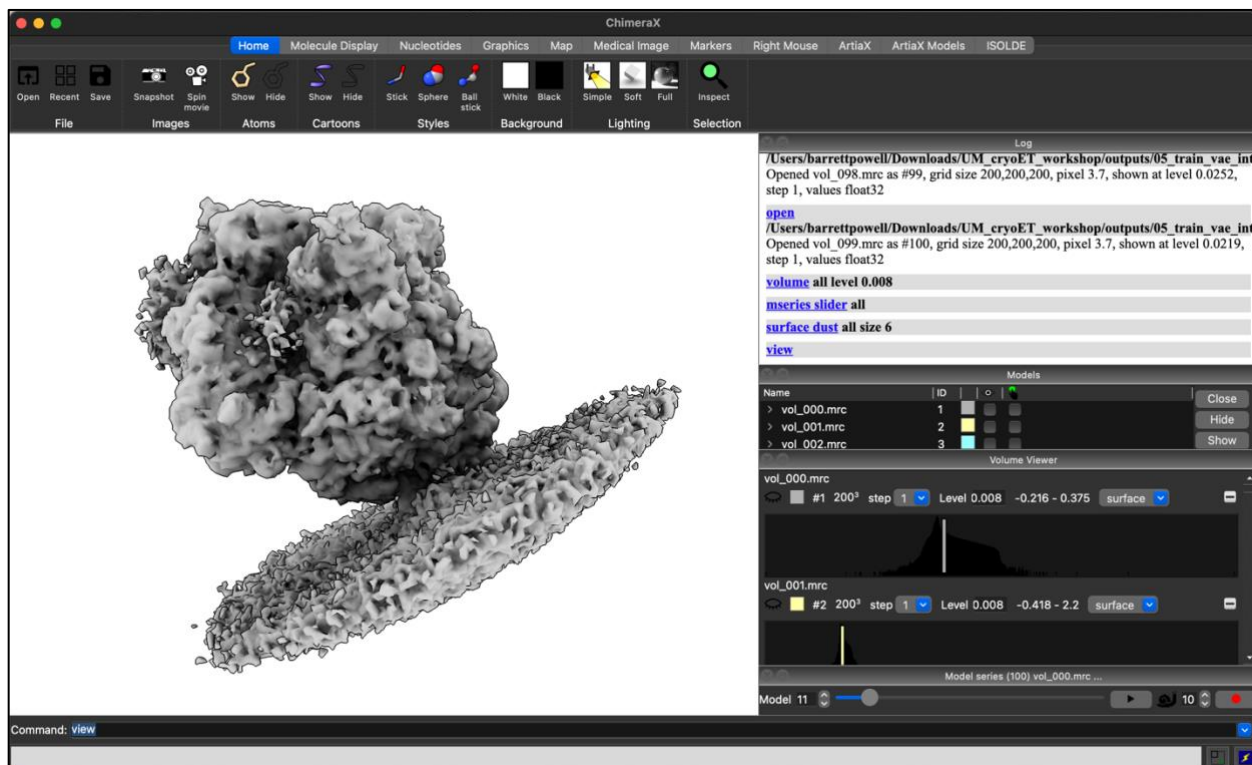
Identify biologically interesting states by k100 visual inspection

ChimeraX

```
open "05_train_vae_intermol_filtered/analyze.49.published/kmeans100/*.mrc"
volume all level 0.008
surface dust all size 6
mseries slider all
```

visually inspect for biologically interesting states

- my k100 labels for membrane-bound ribosome
8, 9, 10
- my k100 labels for E-site / 5' disome
25, 26, 27, 28, 29, 81, 84, 85, 87, 89, 90, 92
- my k100 labels for A-site / 3' disome
30, 31, 34, 35, 39, 42, 45
- my k100 labels for trisome
24, 36, 37, 38, 88



Generate membrane-bound ribosome indices from k100 annotation

Jupyter notebook

Open in jupyter

05_train_vae_intermol_filtered/analyze.49/tomoDRGN_viz+filt.ipynb

When reading in kmeans100, update path to:

```

kmeans_labels = utils.load_pk(f'{WORKDIR}/analyze.{EPOCH}.published/kmeans{K}/labels.pkl')
kmeans_centers = np.loadtxt(f'{WORKDIR}/analyze.{EPOCH}.published/kmeans{K}/centers.txt')
  
```

When reading in volumeseries star file, set path to:

```

path_to_volseries_star = '/data/EMPIAR-11843/data/starfiles/10499_22k_box64_angpix6_volumeseries.star'
  
```

When selecting particles based on k-means clustering, set clusters to:

```

cluster_ids = [8, 9, 10]
  
```

Run the “select particles based on GMM cluster” and “select particles based on z-norm outliers” sections to try other ways to classify particles based on latent embeddings. When finished, re-run the “select particles based on k-means clustering” section to make sure those selections are carried forward. The important thing is to make sure that the cell in the screenshot above is run last before proceeding to “interactive visualization” section below, as this sets the `cluster_ids` variable correctly.

Run the “interactive visualization” first few cells to produce `df_merged`, then skip “interactive selection”, “view tilt images from selected particles”, and “View particle distributions in tomogram context” sections (do not have tomograms available)

Update the path naming the indices to save to:

```
SAVE_PATH = f'{WORKDIR}/ind_keep.{len(ind_keep)}_particles_memribo.pkl'
```

Skip saving non-selected particles

Do not close the notebook yet :)

Terminal:

```
$ python
>>> import numpy as np
>>> from tomodrgrn import utils
>>> k100_labels =
utils.load_pkl('05_train_vae_intermol_filtered/analyze.49.published/kmeans100/labels.pkl')
>>> labels_memribo = [8, 9, 10]
>>> ind_memribo = [ind for ind, label in enumerate(k100_labels) if label in
labels_memribo]
>>> ind_memribo = np.array(ind_memribo)
>>> ind_reindexing = utils.load_pkl('03_train_vae/ind_keep.20981_particles.pkl')
>>> ind_memribo_reindexed = ind_reindexing[ind_memribo]
>>> utils.save_pkl(ind_memribo, '05_train_vae_intermol_filtered/
ind_memribo.482_particles.pkl')
>>> exit()
```

13. Validate membrane-associated ribosomes

Purpose

Just as we validated the particles annotated as 50S in tomoDRGN by a homogeneous RELION reconstruction, we can perform a similar reconstruction of the particles annotated as membrane-associated ribosomes to validate the membrane features.

Filter volume series star file by membrane-associated particle indices

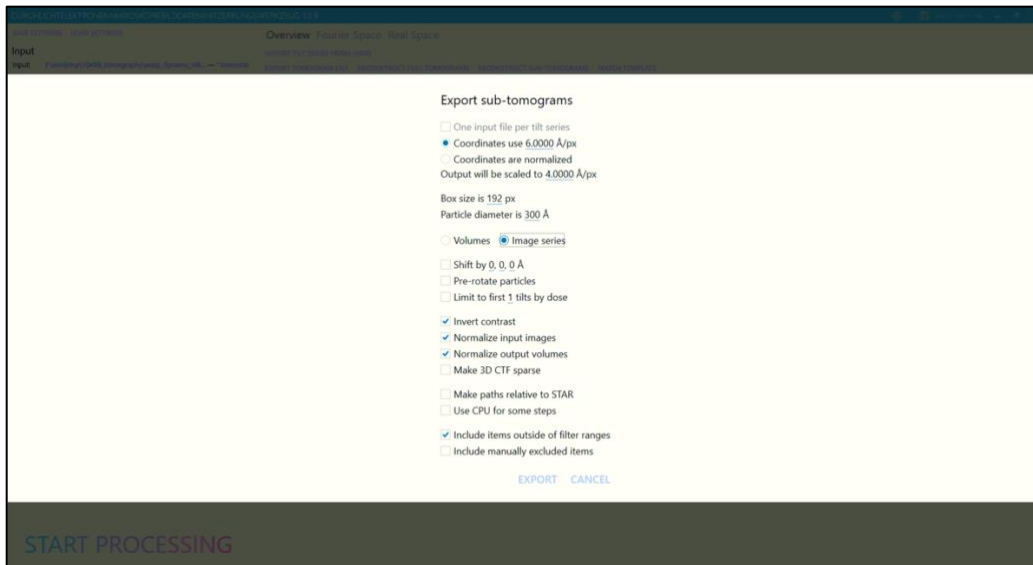
Terminal

```
$ tomodrpn filter_star \  
  /data/EMPIAR-11843/data/starfiles/10499_22k_box64_angpix6_volumeseries.star \  
  --ind 05_train_vae_intermol_filtered/ind_keep.482_particles_memribo.pkl \  
  --ptcl-id-col index \  
  -o  
  05_train_vae_intermol_filtered/10499_22k_box64_angpix6_volumeseries_428_particles_me  
  mribo.star
```

Re-extract memribo particles at intermolecular scale in Warp/M

While we could directly perform a reconstruction of the 50S ribosomes because we had previously extracted all particles as volume series subtomograms with box size 64px and pixel size 6Å/px, this real space box size (384Å) is significantly smaller than that used to train our intermolecular model (200px * 3.7Å/px = 740 Å). Therefore, we need to re-extract the membrane bound ribosome particles as volume series with a larger real space box to properly validate this structure.

It turns out that a subset of the membrane-associated particles exhibit a globular density on the ribosome-distal side of the membrane. As described in the tomoDRGN methods paper, we identify this subset of membrane-associated ribosomes as being in complex with SecDF. This SecDF⁺ subset of particles was re-extracted in Warp as volume series subtomograms with box size 192px and pixel size 4Å/px, following the steps described above to extract particles from Warp/M. We can use this subset of the membrane-associated ribosomes to backproject and validate our membrane-associated ribosome annotations.



Validate isolated particles with traditional tools

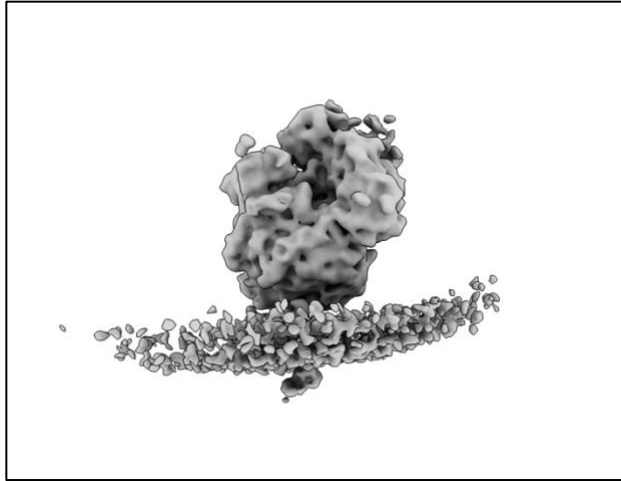
Terminal

```
$ mkdir -p 05_train_vae_intermol_filtered/validation/backproject_memribo
```

```
$ cp /data/EMPIAR-11843/data/starfiles/10499_22k_box192_angpix4_volumeseries_secdfribos.star 05_train_vae_intermol_filtered/validation/backproject_memribo/
```

```
$ sed -i 's/./\subtomo_box192_angpix4_train30_kmeans100_secdfribo_380ptcls/\data\EMPIAR-11843\data\subtomo_box192_angpix4_train30_kmeans100_secdfribo_380ptcls/g' 05_train_vae_intermol_filtered/validation/backproject_memribo/10499_22k_box192_angpix4_volumeseries_secdfribos.star
```

```
$ mpirun -n 9 relion_reconstruct_mpi \
--i
05_train_vae_intermol_filtered/validation/backproject_memribo/10499_22k_box192_angpix4_volumeseries_secdfribos.star \
--o
05_train_vae_intermol_filtered/validation/backproject_memribo/10499_22k_box192_angpix4_volumeseries_secdfribos_reconstruct.mrc \
--3d_rot \
--ctf \
--maxres 15
Timing: 1 minute
```

14. Map tomoDRGN-generated intermolecular volumes to tomogram spatial context

Purpose

While we have thus far examined structural heterogeneity of individual particles in isolation, additional analyses and insights can be gained from mapping heterogeneous volumes to their locations in the source tomogram (here, inside bacterial cells).

The overall process illustrated here is to (1) isolate the latent embeddings associated with the particles from a single tomogram, (2) generate the volumes represented by those embeddings with the trained tomoDRGN model, and (3) prepare a ChimeraX command file that repositions and reorients each volume according to its tomogram-level coordinates and pose.

Visualizing volumes in tomogram context colored by structural heterogeneity class

Jupyter notebook

Return to the [05_train_vae_intermol_filtered Jupyter notebook](#)

At the bottom of the notebook, add and run the following:

```
tomo_ids = (df_merged['_rlnGroupName'].str.split('_').str[0] + '_').unique()
print(f'Unique tomogram identifiers in _rlnGroupName column: {tomo_ids}')
z_cols = [f'z{i}' for i in range(z.shape[1])]
for tomo in tomo_ids:
    SAVE_PATH = f'{WORKDIR}/z.{EPOCH}.{tomo}.pkl'
    tomo_rows = df_merged['_rlnGroupName'].str.contains(tomo)
    z_out = df_merged[tomo_rows][z_cols].to_numpy()
    utils.save_pkl(z_out, SAVE_PATH)
    print(f'Wrote {os.path.abspath(SAVE_PATH)}')

    SAVE_PATH = f'{WORKDIR}/labels.{tomo}.pkl'
    labels = np.zeros(len(df_merged[tomo_rows]))
    labels[df_merged[tomo_rows]['kmeans_labels'].isin([8, 9, 10])] = 1
    labels[df_merged[tomo_rows]['kmeans_labels'].isin([25, 26, 27, 28, 29, 81, 84, 85, 87, 89, 90, 92])] = 2
    labels[df_merged[tomo_rows]['kmeans_labels'].isin([30, 31, 34, 35, 39, 42, 45])] = 3
    labels[df_merged[tomo_rows]['kmeans_labels'].isin([24, 36, 37, 38, 88])] = 4
    utils.save_pkl(labels, SAVE_PATH)
    print(f'Wrote {os.path.abspath(SAVE_PATH)}')
```

This will save the latent embeddings for all particles into separate files by tomogram. It also saves a file specifying a class label for each particle, here defined as which k100 classes the particle belonged to.

Terminal

```
$ tomodrgrn eval_vol \  
  -w 05_train_vae_intermol_filtered/weights.49.pkl \  
  -c 05_train_vae_intermol_filtered/config.pkl \  
  -o 05_train_vae_intermol_filtered/analyze.49.published/tomo00256_vols \  
  --zfile 05_train_vae_intermol_filtered/z.49.00256_.pkl \  
  --downsample 64 \  
  --Apix 11.6
```

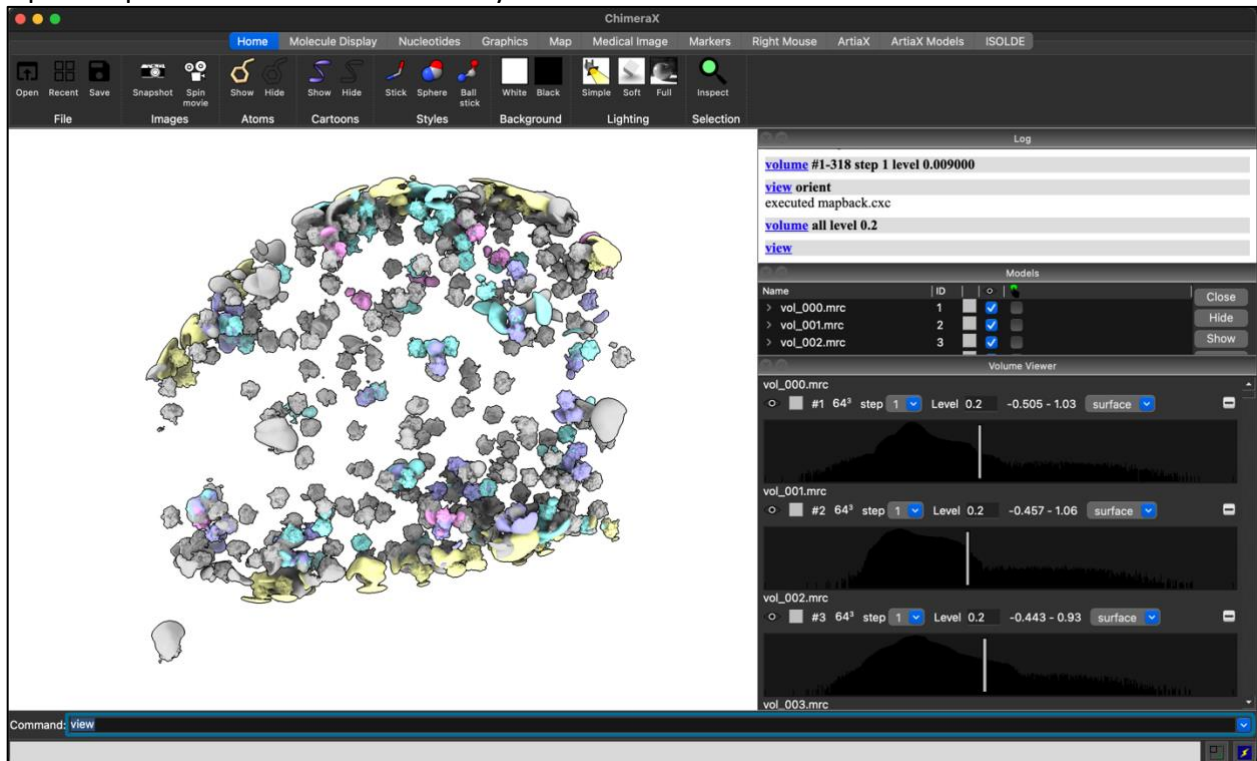
Timing: 1 minute

This command generates all of the volumes from the specified z.pkl file using the specified model weights and volume downsampling settings.

```
$ tomodrgrn subtomo2chimerax \  
  /data/EMPIAR-11843/data/starfiles/10499_22k_box64_angpix6_volumeseries.star \  
  --tomoname 00256.tomostar \  
  --vols-dir 05_train_vae_intermol_filtered/analyze.49.published/tomo00256_vols \  
  --ind 03_train_vae/ind_keep.20981_particles.pkl \  
  --vols-render-level 0.2 \  
  --coloring-labels 05_train_vae_intermol_filtered/labels.00256_.pkl \  
  -o 05_train_vae_intermol_filtered/analyze.49.published/tomo00256_vols/mapback.cxc \  
  --star-apix-override 6 \  
  --vols-apix-override 11.6
```

Chimerax

Open mapback.cxc in ChimeraX and fly around!



```
mapback_rgba_labels.txt x
1 Mapping of unique labels : RGBA specification : chimerax models
2
3 Label : 0.0
4 RGB : [75 75 75 1]
5 models : 1,2,3,4,5,7,8,9,10,12,13,14,15,17,18,19,20,21,22,23,25,26,29,32,33,34,36,37,39,40,41,42,47,48,50,52,53,55,57,58
,60,62,63,64,65,67,68,69,72,73,74,77,78,80,81,82,83,87,88,91,92,94,95,96,97,100,101,102,104,105,106,108,110,111,112,114,
115,118,120,122,125,126,127,128,130,131,134,136,137,138,139,140,141,142,143,144,145,150,151,153,155,157,158,159,160,161,
162,163,165,166,167,168,169,170,174,176,177,183,184,185,186,187,188,189,192,193,194,197,198,200,203,204,205,206,209,211,
212,213,214,215,216,217,218,219,220,222,223,224,226,227,229,230,231,233,235,237,238,241,242,247,250,252,253,255,256,257,
259,261,262,264,265,266,268,269,270,271,273,274,275,276,278,282,283,284,285,287,290,291,292,293,296,297,298,299,301,303,
308,310,311,312,313,314,315,318
6
7 Label : 1.0
8 RGB : [100 100 70 1]
9 models : 24,27,28,38,49,54,59,61,66,71,85,90,93,99,107,109,116,117,129,132,135,146,149,152,156,180,240,248,258,267
10
11 Label : 2.0
12 RGB : [ 70 100 100 1]
13 models : 6,16,30,31,46,56,70,79,84,86,89,98,113,119,133,147,154,171,172,179,181,182,190,195,196,201,207,208,210,221,236,
239,243,245,246,251,254,260,263,272,277,279,281,286,295,302,306,317
14
15 Label : 3.0
16 RGB : [ 70 70 100 1]
17 models : 11,35,44,45,51,75,76,103,121,124,164,173,175,191,225,228,232,234,249,280,288,289,294,304,305,307,309
18
19 Label : 4.0
20 RGB : [100 70 100 1]
21 models : 43,123,148,178,199,202,244,300,316
22
```

The `subtomo2chimerax` command also produces another file:

`05_train_vae_intermol_filtered/analyze.49.published/tomo00256_vols/mapback_rgba_labels.txt` . This plain text file is shown above, and lists each of the classes that we labeled in the Jupyter notebook at the beginning of this section: their numerical index, the RGB color associated with that index (in %red, %green, %blue, alpha), and the 1-indexed ChimeraX models associated with each volume. This is designed such that you can more easily interpret which volumes belong to which color-coded class, and that you can select all volumes in a particular class.

15. Map tomoDRGN-generated intramolecular volumes to tomogram spatial context

Purpose

We can map back tomoDRGN volumes from the intramolecular model we trained earlier when separating 50S from 70S particles. Generating and mapping these volumes back to tomogram locations allows us to explore different spatial distributions of, for example, 50S vs 70S ribosomes.

Visualizing volumes in tomogram context colored by structural heterogeneity class

Jupyter notebook

[Return to the 04_train_vae_filtered Jupyter notebook](#)

At the bottom of the notebook, add and run the following:

```
tomo_ids = (df_merged['_rlnGroupName'].str.split('_').str[0] + '_').unique()
print(f'Unique tomogram identifiers in _rlnGroupName column: {tomo_ids}')
z_cols = [f'z{i}' for i in range(z.shape[1])]
for tomo in tomo_ids:
    SAVE_PATH = f'{WORKDIR}/z.{EPOCH}.{tomo}.pkl'
    tomo_rows = df_merged['_rlnGroupName'].str.contains(tomo)
    z_out = df_merged[tomo_rows][z_cols].to_numpy()
    utils.save_pkl(z_out, SAVE_PATH)
    print(f'Wrote {os.path.abspath(SAVE_PATH)}')

    SAVE_PATH = f'{WORKDIR}/labels.{tomo}.pkl'
    labels = np.zeros(len(df_merged[tomo_rows]))
    labels[df_merged[tomo_rows]['kmeans_labels'].isin([2, 3, 4, 5, 7, 9])] = 1
    utils.save_pkl(labels, SAVE_PATH)
    print(f'Wrote {os.path.abspath(SAVE_PATH)}')
```

Terminal

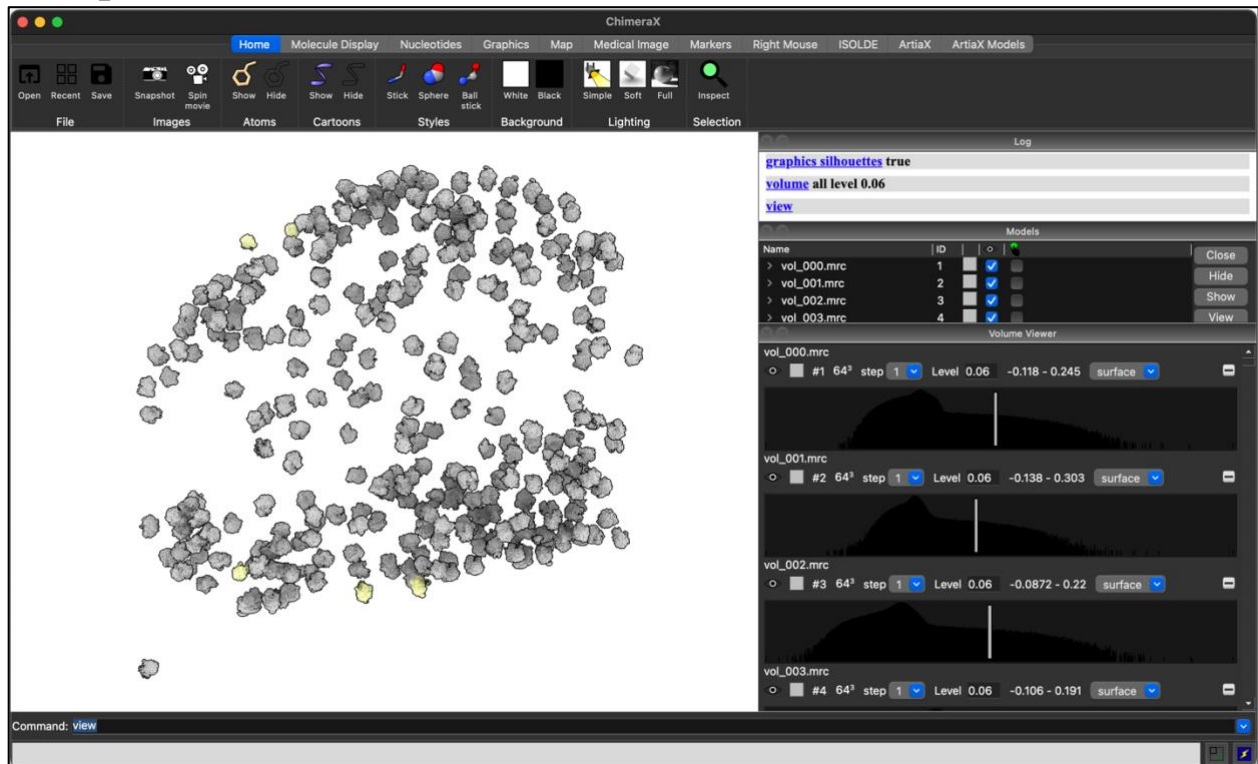
```
$ tomodrgn eval_vol -w outputs/04_train_vae_filtered/weights.49.pkl \
  -c outputs/04_train_vae_filtered/config.pkl \
  -o outputs/04_train_vae_filtered/analyze.49.published/tomo00256_vols \
  --zfile outputs/04_train_vae_filtered/z.49.00256_.pkl \
  --downsample 64 \
  --Apix 5.5565

$ tomodrgn subtomo2chimerax \
  data/starfiles/10499_22k_box64_angpix6_volumeseries.star \
  --tomoname 00256.tomostar \
  --vols-dir outputs/04_train_vae_filtered/analyze.49.published/tomo00256_vols \
  --ind outputs/03_train_vae/ind_keep.20981_particles.pkl \
  --vols-render-level 0.06 \
  --coloring-labels outputs/04_train_vae_filtered/labels.00256_.pkl \
  -o outputs/04_train_vae_filtered/analyze.49.published/tomo00256_vols/mapback.cxc \
```

```
--star-apix-override 6 \  
--vols-apix-override 5.5565
```

Chimerax

Open mapback.cxc in ChimeraX and fly around! Not a lot of 50S ribosomes in this tomogram; try with TS_301



What might come next?

- Systematic interrogation of structural heterogeneity across large volume ensembles (k100, k500, k1000, k1000, ..., all volumes)
 - <https://github.com/lkinman/MAVEn>
 - (Atomic)-model guided analysis of volume density (aka occupancy)
 - Is there correlated heterogeneity of structural blocks? Which particle subsets exhibit these structural features?
 - Literature: <https://www.nature.com/articles/s41594-023-01078-5>
 - <https://github.com/lkinman/SIREn>
 - Model-free detection of correlated structural elements from a volume ensemble
 - What types of compositional heterogeneity are present in this volume ensemble? Conformational heterogeneity?
 - Literature: manuscript in preparation
 - <https://phenix-online.org/documentation/reference/varref.html>
 - Refinement of an atomic model into an ensemble of maps
 - What questions can I better answer by parameterizing structural heterogeneity with an atomic model ensemble?
 - Literature: <https://doi.org/10.1016/j.bbamem.2023.184133>
- Refinements of isolated particle sets to confirm observed structural heterogeneity and resolve to better resolution
 - Relion 3D auto-refine
 - M multispecies refinement (perhaps each species corresponds to a tomoDRGN-separated distinct structural state)
- Your own custom downstream analysis!

Frequently asked questions

- What types of structural heterogeneity can tomoDRGN learn?
 - TomoDRGN’s design provides minimal constraints to the type of structural heterogeneity that can be learned. Generally, tomoDRGN will learn any features present in the input images: proteins, RNA, membranes, etc. This means tomoDRGN can theoretically learn both compositional and conformational heterogeneity.
- How many epochs should I train my tomoDRGN model for? What do over/under fitting look like?
 - We have generally observed that models are well trained between 25 – 50 epochs of training, across different types of particles and dataset sizes spanning 500 – 25000 particles. An undertrained model may produce volumes that look low resolution, homogeneous to each other, and potentially with artifacts like spikes of density along the orthogonal axes. An overtrained model may produce a latent space that initially contained distinct clusters but which have now merged, and volumes that appear heavily oversharpened and may have spurious density scattered throughout the box. We generally recommend to perform model analysis on the first epoch at which the latent space and volume space appear to have stabilized (note that this often does not correlate with a plateau in the loss curve).
- My particles have been processed in [STOPGAP / RELION v4/v5 / Warp v2 / etc]. How can I use them with tomoDRGN?
 - TomoDRGN was initially developed to work with Warp and M (version 1). We are actively working to expand the set of directly compatible STA software, but for the moment this means particles input to tomoDRGN must be exported from Warp or M as “particle series” subtomograms.
- My particle is symmetric, does that change how I should use tomoDRGN?
 - TomoDRGN’s voxel-wise decoder module was not designed with symmetric particles or symmetry operators in mind. We would like to implement this, but it is not at the top of our current priority list. We recommend performing symmetry relaxation to C1 (demonstrated for apoferritin in EMPIAR-10491 and for HIV Gag in EMPIAR-10164), or symmetry expansion with signal subtraction to the “asymmetric unit” of the symmetric particle.
- What resolution do I need to have by STA for tomoDRGN to be useful?
 - We frequently use tomoDRGN early in good / bad particle filtration at resolutions of up to ~10 to ~15 Å. Large compositional heterogeneity can be distinguished around these resolutions. Finer compositional or conformational heterogeneity usually requires better resolutions. Generally tomoDRGN should be viewed as

tool to observe and generate hypotheses relating to structural heterogeneity, to then be orthogonally validated and tested.

- How many particles do I need for tomoDRGN to be useful?
 - We have run tomoDRGN and extracted useful results with datasets as small as ~500 particles (see Figure 6e of <https://doi.org/10.1038/s41592-024-02210-z>). As mentioned above, these results should serve as hypothesis generating tools and should be further validated.
- How large do particles need to be for tomoDRGN to be useful?
 - We have used tomoDRGN on datasets of purified or partially purified complexes spanning 500 – 3000 kDa, and on datasets of ribosomes *in situ*. We would love to hear about other use cases of tomoDRGN!
- Does tomoDRGN refine particle poses for each structurally heterogeneous state?
 - No, tomoDRGN does not refine particle poses. It can be thought of as (a very powerful per-particle version of) 3-D classification without pose optimization.
- I don't see any meaningful heterogeneity with tomoDRGN, what should I do?
 - There are a few possible explanations for not seeing heterogeneity as the output of tomoDRGN.
 - Your model may be undertrained (in which case, train additional epochs) or may have too small model capacity (in which case, train a new model with a larger encoder, latent, and decoder dimensionality).
 - Your input dataset may be too low resolution to distinguish features (in which case, try to improve your particle count and resolution by STA).
 - Your dataset may have too weak of signal for tomoDRGN to see, perhaps due to low molecular weight particles, significant disorder in the particles, substantial background signal, etc. These problems likely require alternative sample preparation and dataset collection strategies to mitigate.
 - Your particle may exhibit heterogeneity of too small an amplitude for tomoDRGN to detect (e.g. small shifts of individual loops).
 - Your dataset may exhibit purely conformational heterogeneity which may benefit from the “conservation of mass” regularization enforced by heterogeneity analysis tools explicitly designed to resolve conformational heterogeneity (TomoFlow, HEMNMA-3D, DeepHEMNMA, MDTOMO, etc).
 - Your sample may contain no structural heterogeneity to speak of.
- How does tomoDRGN's classification of “junk” particles compare to 3D / 2D classification tools?
 - Due to the extremely low SNR and absence of ground truth labels in experimental datasets, it is extremely difficult to perform absolute performance comparisons between different algorithmic approaches. However, in our hands,

tomoDRGN appears to be very effective at identifying and separating “junk” particles. This is likely due to the highly expressive multidimensional latent space and per-particle generative model as compared with enforcing all particles to be sorted into k distinct classes.

Supplemental: voxel_pca_umap.py

```
'''
Runs real space PCA on all volumes provided, runs UMAP on the first `--num-pcs` PCs, saves both results
to a pkl
Also plots and saves PCA relative variance, first 5 PCs against each other, and UMAP1 vs UMAP2
'''

import argparse
import os
import glob
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.decomposition import PCA
import umap
from tomodrgrn import mrc, utils

def add_args(parser):
    parser.add_argument('--vol-dir', type=os.path.abspath, required=True, help='path to directory
containing volumes to analyze')
    parser.add_argument('--out-dir', type=os.path.abspath, required=True, help='path to directory to save
outputs')
    parser.add_argument('--num-pcs', type=int, default=128, help='keep this many PCs when saving PCA
and running UMAP')
    parser.add_argument('--mask-type', choices=['none', 'spherical'], default='spherical', help='binary real-
space mask to apply to each volume')
    parser.add_argument('--densmap', action='store_true', help='use the DensMAP flag to better preserve
local density during UMAP reduction')

    return parser

def main(args):
    # SETUP: assert vol-dir exists and is not empty, create outdir
    print('Validating inputs ...')
    assert os.path.isdir(args.vol_dir)
    os.makedirs(args.out_dir, exist_ok=True)

    # PREPROCESSING: create natural-sorted list of volumes to iterate through
    print('Finding volumes ...')
    vols_list = glob.glob(os.path.join(args.vol_dir, '*.mrc'))
    vols_list.sort(key=lambda x: int(os.path.basename(x).split('_')[-1].split('.mrc')[0])) # assumes naming
format `vol_001.mrc`
    box_size = mrc.parse_mrc(vols_list[0])[0].shape[0]

    # PREPROCESSING: prepare mask
```

```

print(f'Preparing mask of type {args.mask_type} ...')
if args.mask_type == 'spherical':
    xx = np.linspace(-1, 1, box_size, endpoint=True if box_size % 2 == 1 else False)
    z, y, x = np.meshgrid(xx, xx, xx)
    coords = np.stack((x, y, z), -1)
    r = np.sum(coords ** 2, axis=-1) ** 0.5
    mask = np.where(r > 1, 0, 1).flatten().astype(bool)
elif args.mask_type == 'none':
    mask = np.ones((box_size, box_size, box_size)).flatten().astype(bool)
else:
    raise RuntimeError

# PREPROCESSING: load volumes
print('Loading and masking volumes ...')
vols = np.zeros((len(vols_list), np.sum(mask)), dtype=np.float32)
for i, vol in enumerate(vols_list):
    vol_unmasked = mrc.parse_mrc(vol)[0].flatten()
    vols[i] = vol_unmasked[mask]

# PROCESSING: run PCA, keep first num_pcs, save pkl
print('Running PCA ...')
assert args.num_pcs <= vols.shape[1]
pca = PCA(n_components=args.num_pcs, random_state=42, copy=False)
pc = pca.fit_transform(vols)
utils.save_pkl(pc, os.path.join(args.out_dir, 'voxel_pc.pkl'))

# PLOTTING: plot and save relative PC variance
print('Plotting explained variance ratio ...')
x = np.arange(args.num_pcs)
y = pca.explained_variance_ratio_
fig, ax = plt.subplots(1, 1)
ax.bar(x, y)
ax.set_xlabel('principal components')
ax.set_ylabel('explained variance ratio')
plt.tight_layout()
plt.savefig(os.path.join(args.out_dir, 'voxel_pc_explained-variance-ratio.png'))
plt.close()

# PLOTTING: plot and save first 4 PCs against each other
print('Plotting first several PCs ...')
max_n_pcs_plotted = 4
for i in range(max_n_pcs_plotted - 1):
    for j in range(i + 1, max_n_pcs_plotted):
        x = pc[:, i]
        y = pc[:, j]
        fig, ax = plt.subplots(1, 1)
        if len(x) < 500:
            ax.scatter(x, y, s=0.1)

```

```

else:
    sns.jointplot(x=x, y=y, kind='kde', space=0)
    ax.set_xlabel(f'v-PC{i+1}')
    ax.set_ylabel(f'v-PC{j+1}')
    plt.tight_layout()
    plt.savefig(os.path.join(args.out_dir, f'voxel_pc_pc{i+1}-pc{j+1}.png'))
    plt.close()

# PROCESSING: run UMAP, save pkl
print(f'Running UMAP on first {args.num_pcs} PCs ...')
reducer = umap.UMAP(densmap=args.densmap, random_state=42)
embedding = reducer.fit_transform(pc)
utils.save_pkl(embedding, os.path.join(args.out_dir, 'voxel_pc_umap.pkl'))

# PLOTTING: plot and save UMAP1 vs UMAP2
print('Plotting UMAP ...')
x = embedding[:, 0]
y = embedding[:, 1]
fig, ax = plt.subplots(1, 1)
if len(x) < 500:
    ax.scatter(x, y, s=0.1)
else:
    sns.jointplot(x=x, y=y, kind='kde', space=0)
    ax.set_xlabel('v-UMAP1')
    ax.set_ylabel('v-UMAP2')
    plt.tight_layout()
    plt.savefig(os.path.join(args.out_dir, 'voxel_pc_umap1-umap2.png'))
    plt.close()

if __name__ == '__main__':
    parser = argparse.ArgumentParser(description=__doc__,
    formatter_class=argparse.ArgumentDefaultsHelpFormatter)
    add_args(parser)
    main(parser.parse_args())

```