AlphaCryo4D v0.1.0b Documentation

AlphaCryo4D is an open-source free software released under GNU General Public LICENSE that implements 3D classification of single-particle cryo-EM data using deep manifold learning and novel energy-based particle voting methods.

AlphaCryo4D v0.1.0b is currently a prototype, which is still under development. Users are free to use and modify the source code, providing their compliance with the GPL and that any publication making use of this software shall cite the following reference or its formally published form:

Zhaolong Wu, Shuwen Zhang, Wei Li Wang, Yinping Ma, Yuanchen Dong, Youdong Mao. Deep manifold learning reveals hidden dynamics of proteasome autoregulation. bioRxiv preprint doi: https://doi.org/10.1101/2020.12.22.423932.

Installation

1. Create the conda environment:
conda create -n AlphaCryo4D python=3.7.1

2. Activate the environment:
source activate AlphaCryo4D

3. Install the dependencies:
conda install --yes --file EnvConda.txt
pip install -r EnvPip.txt

Programs and Scripts

Bootstrap Directory: 3D bootstrap with M-fold data augmentation and particle shuffling

1. randsf.py
Purpose: Split the star file of all particles into several sub-datasets.

Usage:
python Bootstrap/randsf.py --star $star_fn --number $nr_prtl

Control parameters:
--star, -s <particles.star>
    Star file of all particles
--number, -n <100000>
    Particle number of each sub-dataset

2. bootstrap.py
Purpose: Augment each sub-dataset to M times.

Usage:
python Bootstrap/bootstrap.py --star $batch_fn --fold $m_fold

Control parameters:
--star, -s <batch1.star>
        Star file of a batch of dataset
--fold, -f <3>
        Fold of data augmentation

3. link.sh
Purpose: Link the star and mrc files of 3D bootstrapping into two folders.

Usage:
bash Bootstrap/link.sh

Control parameters:
Set the fold identifier ur, batch number br, class number cr and your path of star and mrc files

4. fit.sh
Purpose: Align all 3D density maps to a reference map.

Usage:
bash Bootstrap/fit.sh

Control parameters:
Set your path of the reference map

5. bigdata.py
Purpose: Prepare the 3D volume dataset after bootstrapping.

Usage:
python Bootstrap/bigdata.py --folder $maps_path

Control parameters:
--folder, -f <./maps_aligned/>
        Folder of maps

DeepFeature Directory: extract 3D feature by 3D deep residual autoencoder

6. run_prepare.py
Purpose: Preprocess the input data for deep neural network.

Usage:
python DeepFeature/run_prepare.py --data $data_fn

Control parameters:
--data, -d <rdata_3d.npy>
        3D data stack

7. run_resnet.py
Purpose: Train the 3D residual autoencoder network.

Usage:
python DeepFeature/run_resnet.py --epoch $nr_epoch --batchsize $batch_size --validationsize $validation_size --regularization $coef_reg --data $input_fn --checkpoint $check_fn --finalmodel $finial_fn --gpu $gpu

Control parameters:
--epoch, -e <50>
    Training epochs
--batchsize, -b <8>
    Batch size of training
--validationsize, -v <800>
    Size of validation data
--regularization, -r <0>
    L2 regularization coefficient
--data, -d <data_dl.npy>
    Input 3D data
--checkpoint, -c <checkpoint/check.h5>
    Path of best model
--finalmodel, -f <model/final_model.h5>
    Path of final model
--gpu, -g <0,1,2,3>
    Gpu utilization

8. run_predict.py
Purpose: Calculate the 3D feature by the autoencoder.

Usage:
python DeepFeature/run_predict.py --data $input_fn --model $model_fn --batchsize $batch_size --feature $feature_fn

Control parameters:
--data, -d <data_dl.npy>
    Input 3D data
--model, -m <checkpoint/check.h5>
    Model of network for feature extraction
--batchsize, -b <8>
    Batch size of prediction
--feature, -f <result/feature.npy>
    Output of feature engineering

ManifoldLandscape Directory: *manifold embedding of 3D volume data for free-energy landscape reconstitution*

9. tsne_prepare.py
Purpose: Preprocess the 3D density maps and their feature maps for manifold learning.

Usage:
python ManifoldLandscape/tsne_prepare.py --data $data_fn --feature $feature_fn

Control parameters:
--data, -d <rdata.npy>
	Raw data
--feature, -f <result/feature.npy>
	Deep feature

10. tsne_rd.py
Purpose: Map the two-dimensional manifold embedding of input volume data by t-SNE.

Usage:
python ManifoldLandscape/tsne_rd.py --input $input_fn --output $output_fn --seed $random_seed --perplexity $perplexity --niter $niter

Control parameters:
--input, -i <input.npy>
	Input of t-SNE
--output, -o <output.npy>
	Output of t-SNE
--seed, -s <0>
	Random seed
--perplexity, -p <30.0>
	Perplexity of t-SNE
--niter, -n <1000>
	Maximum number of iterations in t-SNE

11. enumerate.sh
Purpose: Count the particle number of each data point.

Usage:
bash ManifoldLandscape/enumerate.sh

12. string_method.py
Purpose: Compute the free energy landscape and find the minimum energy path by the string method.

Usage:
python ManifoldLandscape/string_method.py --landscape $tsne_fn --number $num_fn --range $range --start $start_point --end $end_point --stepmax $max_step --stepsize $step_size --tolerance $tolerance --interpolate $interpolate --kcenters $nr_centers

Control parameters:
--landscape, -l <output.npy>
	Free energy landscape points mapping by t-sne
--number, -n <num.txt>
	Particle number file
--range, -r <100.0>
	Maximum value of the reaction coordinate of free energy landscape

--start, -s <-80.0 -80.0>
Position around start point of transition path
--end, -e <80.0 80.0>
Position around end point of transition path
--stepmax, -m <100000>
Maximum steps in String method algorithm
--stepsize, -i <1e-1>
Step size in String method algorithm
--tolerance, -t <1e-7>
Convergence condition in String method algorithm
--interpolate, -I <linear>
Interplotion method used in free energy landscape plotting, linear or cubic
--kcenters, -k <20>
K clustering centers along transition path

ParticleVoting Directory: 3D classification with particle voting on the free energy landscape

13. clustering.py
Purpose: Find the ID and distance to the nearest clustering center of the data points within one cluster.

Usage:
python ParticleVoting/clustering.py --points $tsne_fn --centers $centers_fn --radius $radius

Control parameters:
--points, -p <output.npy>
Files of data points
--centers, -c <centers_k.npy>
Files of clustering centers
--radius, -r <30.0>
Clustering radius

14. vote_prepare.sh
Purpose: Prepare the star files of the points of one cluster.

Usage:
bash ParticleVoting/vote_prepare.sh $c

Control parameters:
Set the ID and distance file $c and your path of star files

15. gethead.py
Purpose: Get the header lines of these star files.

Usage:
python ParticleVoting/gethead.py --star $star_fn --head $head_fn

Control parameters:

--star, -s <u123_p1c1.star>
Input star file
--head, -he <head.star>
Head of the star file

16. post_vote_and.sh
Purpose: Vote on particles by the energy-based algorithm.

Usage:
bash ParticleVoting/post_vote_and.sh

Control parameters:
Set the threshold of voting th

17. post_and_f.sh
Purpose: Classify particles quickly by the energy-based particle-voting algorithm.

Usage:
bash ParticleVoting/post_and_f.sh

Control parameters:
Set the threshold of voting th

18. post_or_parallel.sh
Purpose: Classify particles by the distance-based algorithm in parallel.

Usage:
bash ParticleVoting/post_or_parallel.sh

Control parameters:
Set the threshold of voting th and core number nproc

19. dedup.sh
Purpose: Remove the duplication of particles when using the distance-based algorithm.

Usage:
bash ParticleVoting/dedup.sh

Control parameters:
Set the core number nproc