AlphaCryo4D v0.1.1 Tutorial

This tutorial is intended to demonstrate the usage of the current version of AlphaCryo4D by example procedures. The inputs of AlphaCryo4D are cryo-EM particles with the parameters of CTF and alignment. The results of AlphaCryo4D are plots of computed energy landscape and corresponding 3D classes. This tutorial explains examples related to process the dataset of substrate-engaged human 26S proteasome (EMPIAR-10669). The full dataset can be downloaded via the link:
https://www.ebi.ac.uk/empiar/EMPIAR-10669/

The complete analysis of this dataset has been described and discussed in the following reference:

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.

The example scripts, input and output files explained in this tutorial are provided in the folder 'Examples/'. The complete proteasome dataset (EMPIAR-10669) is too large to allow users to run through quick tests for tutorial purpose. Thus, to simplify the demonstration and to allow users to run demo scripts with little efforts and minimal computational resources, the related exemplary datasets, inputs and outputs are reduced to appropriate or smaller sizes to simplify the demonstration and to facilitate testing run in different steps, because it is not possible to include large datasets within this tutorial package due to the storage limitation in Github.com. After users understand how the procedures are executed, they can replace input/data files with their own dataset.

1. Quick start

The tutorial starts in the folder 'Examples/'. This part aims to help users run AlphaCryo4D in a rapid way, with many steps merging together to simplify the whole process. First, the particle star file is split into several batches to perform Bayesian bootstrapping.

cd Fast
python ../../Bootstrap/Star.py --star initial.star --number 100000 --relion 2 --fold 3

The input of this command is the overall star file 'initial.star', containing 200,000 proteasome particles in total (the corresponding particle images are not

provided here due to the storage limitation on github.com). The outputs of this command are several folders started with ‘batch’, with each folder having a star file started with ‘batch’ of 100,000 particles and 4 (fold + 1) star files started with ‘union’. For each ‘union’ star file, the bootstrapped 3D volumes via 3D classification are produced in RELION. The RELION version can be set in this command.

Secondly, RELION is used to perform 3D volume bootstrapping, here we give an example of 3D classification command in RELION:

mkdir –p Class3D/u123

relion_refine_mpi --o Class3D/u123/run --i ./union123.star --particle_diameter 411 --angpix 1.37 --ref maps/refs/ref.mrc --firstiter_cc --ini_high 60 --ctf --ctf_corrected_ref --iter 30 --tau2_fudge 4 --K 10 --flatten_solvent --zero_mask --solvent_mask mask.mrc --oversampling 1 --dont_combine_weights_via_disc --skip_align --strict_highres_exp 10 --sym C1 --norm –scale

Note that some arguments such as ‘--K’, ‘--solvent_mask’ and ‘--strict_highres_exp’ can be adjusted as required. And the files of particle images and masks are not provided here due to the storage limitation. These 4 ‘union’ star files can generate 4 × K volumes (K is the class number set in RELION).

Third, these bootstrapped 3D volumes are used to plot the free energy landscape via deep manifold learning. Given the computational cost, here we use only 3 volumes for demonstration purpose. The volume files and corresponding star files are located in the folder ‘./maps/’ and ‘./stars/’, respectively. These 3D volumes can be aligned to a common reference with the script ‘fit.sh’ before the pseudo-energy landscape calculation. Note that EMAN2 is called in ‘fit.sh’.

. fit.sh

python ../../Auto4D.py --folder maps_aligned/ --stars stars/ --std True --batchsizetrain 2 --validationsize 2 --regularization 0.001 --batchsizepredict 3 --gpu 0,1 --seed 0 --perplexity 30.0 --niter 1000 --landscape output_big.npy --number num_big.txt --range 160 --start -50 100 --end -80 -30 --stepsize 0.1 --interpolate cubic --kcenters 7 --radius 30

The inputs of this step are 3D volume files in the folder ‘./maps/’ with their corresponding star files in the folder ‘./stars/’ and the common reference density map ‘./maps/refs/ref.mrc’. So the 4th and 10th lines of script ‘fit.sh’ should be modified to include this path. The script ‘fit.sh’ prepares the bootstrapped 3D volumes in the folder ‘./maps_aligned/’. The outputs of this step are the merged 3D volume data ‘rdata.npy’ and ‘rdata_3d.npy’ with its list file ‘data.log’, the free

energy landscape ‘el.pdf’, the clustering centers ‘centers_k.npy’, and three folders ‘Deep’, ‘Manifold’ and ‘Vote’. Note that we use the landscape of all 1,000 volumes (‘output_big.npy’ and ‘num_big.txt’ are given) to demonstrate the application of the String method, as three data points are too few for this calculation.

Lastly, particle voting is conducted to purify the particle set. Here we use a small cluster ‘c1_small’ containing the previously mentioned 3 data points instead of ‘Vote/c1’ to demonstrate the process of particle voting.

cd Vote/c1
cp ../../c1_small .
python ../../../../ParticleVoting/Vote.py --classes c1_small --relion 2 --threshold 2

The inputs of this command are the particle sets of the 3D volumes in ‘c1_small’. After setting the voting threshold (M/2 is recommended), the outputs are the head file ‘head.star’, the files both before voting (‘pre_vote.log’ and ‘pre_vote.star’) and after voting (‘post_vote.log’ and ‘post_and.star’). The star file ‘post_and.star’ is what we need.

2. Bootstrapping of 3D volumes

2.1 Particle shuffling

This tutorial starts in the folder ‘Examples/’. To demonstrate how to do particle shuffling, we assume an input star file of 200,000 proteasome particles (the corresponding particle images are not provided here due to the storage limitation on Github.com). First, we expect to split it into 2 batches, with each batch having 100,000 particles. Data with low SNR should contain more particles per batch for bootstrapping.

cd Bootstrap
python ../../Bootstrap/randsf.py --star initial.star --number 100000

The input of this step is the overall star file ‘initial.star’, and the outputs are several star files started with ‘batch’.

Next, for each batch, do a *M*-fold particle shuffling. Here we apply *M* = 3 to the first star file. The smaller the dataset is, the larger value of *M* should be set.

python ../../Bootstrap/bootstrap.py --star batch1.star --fold 3

The input of this step is the star file ‘batch1.star’, and the outputs are 4 star files started with ‘union’. For each ‘union’ star file, the bootstrapped 3D volumes via 3D classification are produced in RELION.

To demo the usage of RELION in this step, here we give an example of 3D classification command in RELION:

mkdir –p Class3D/u123

relion_refine_mpi --o Class3D/u123/run --i ./union123.star --particle_diameter 411 --angpix 1.37 --ref maps/refs/ref.mrc --firstiter_cc --ini_high 60 --ctf --ctf_corrected_ref --iter 30 --tau2_fudge 4 --K 10 --flatten_solvent --zero_mask --solvent_mask mask.mrc --oversampling 1 --dont_combine_weights_via_disc --skip_align --strict_highres_exp 10 --sym C1 --norm –scale

Note that the arguments such as '--K', '--solvent_mask' and '--strict_highres_exp' can be set as required. And the files of particle images and mask are not provided here due to the storage limitation.

2.2 Preprocessing of 3D volumes

These 4 'union' star files can generate 4×K volumes (K is the class number set in RELION). Given the computational cost, here we use only 3 volumes for demonstration purpose. The volume files and corresponding star files are located in the folder './maps/' and './stars/', respectively, under the current path. These 3D volumes can be aligned to a common reference. Note that EMAN2 is called in 'fit.sh'.

. fit.sh

The inputs of this step are 3D volume files in the folder './maps/' and the common reference density map './maps/refs/ref.mrc'. So the 4th and 10th lines of script 'fit.sh' should be modified to include this path. The outputs are the bootstrapped 3D volumes in the folder './maps_aligned/' for the subsequent process.

python ../../Bootstrap/bigdata.py --folder maps_aligned/ --std True

Alternatively, the preprocessing of low-pass filtering (5 Å is recommended) can be conducted instead of standardization by setting '--std False'. The input of this step is the 3D volumes in the folder './maps_aligned/', and the output files are 'rdata.npy' and 'rdata_3d.npy', with their order recorded in 'data.log'. Three typical 3D bootstrapped density maps of proteasome are shown below. The users can check their own bootstrapped 3D volumes using software like UCSF Chimera in this step.

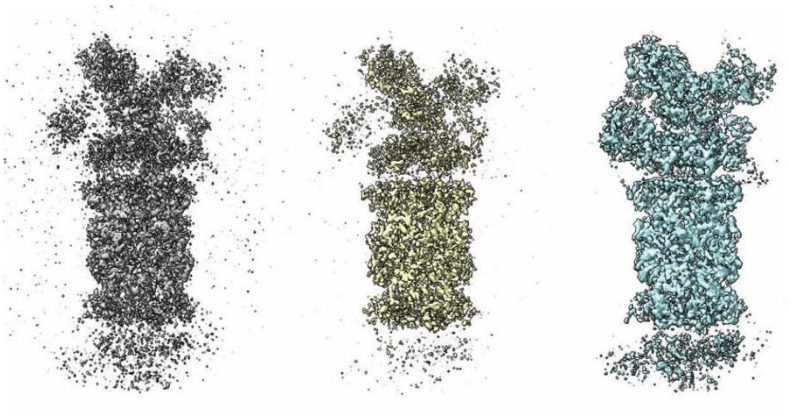

Fig. 1. Typical bootstrapped 3D volumes.

3. Feature extraction by 3D autoencoder

To prepare for the processing of 3D Autoencoder, the dataset of these three 3D volumes should be reformatted firstly.

cd ../DeepFeature
python ../../DeepFeature/run_prepare.py --data ../Bootstrap/rdata_3d.npy

The input of this step is '../Bootstrap/rdata_3d.npy', and the output file is 'data_dl.npy'.

When training the network, we can define the parameters such as the batch size and GPUs according to our requirement. Note that the batch size has to be an integral multiple of the number of GPUs.

python ../../DeepFeature/run_resnet.py --batchsize 2 --validationsize 2 -r 0.001 --data data_dl.npy --gpu 0,1

The input file is 'data_dl.npy', and the outputs of this step are the best model './checkpoint/check.h5', the final model './model/final_model.h5' and the file 'train_process.npz' recording the training loss of each epoch. Then we apply the best model to obtain the feature maps of input 3D volumes.

python ../../DeepFeature/run_predict.py --data data_dl.npy --batchsize 3

The input files of this step are 'data_dl.npy' and './checkpoint/check.h5', and the output is saved to the default path 'Examples/result/feature.npy'.

4. Energy landscape by manifold learning

4.1 Energy landscape

To prepare the input data to compute the energy landscape, the three example 3D volumes and their feature maps are concatenated at first.

cd ../ManifoldLandscape

python ../../ManifoldLandscape/tsne_prepare.py --data ../Bootstrap/rdata.npy --feature ../DeepFeature/result/feature.npy

The inputs are '../Bootstrap/rdata.npy' and '../DeepFeature/result/feature.npy', and the output file of this step is 'input.npy'.

Then t-SNE is used to plot the low-dimension mapping of 3D volumes with a random seed we set. In many cases, the default values of perplexity and maximum number of iterations in t-SNE work well, but can be empirically tuned to improve the results if desired.

python ../../ManifoldLandscape/tsne_rd.py --input input.npy -s 0 --perplexity 30.0 --niter 1000

The input of this step is 'input.npy', and the outputs are the file 'output.npy' of low-dimension mapping and corresponding plot 'tsne.png'. And the particle number of each 3D volume is calculated.

. enumerate.sh

The input of this step is the 3D volumes' star files in the folder '../Bootstrap/stars/'. So note that the 11th line of the script 'enumerate.sh' should refer to this path. The output is the particle number file 'num.txt'. Based on the low-dimensional mapping and the particle number of 3D volumes, we can estimate the energy landscape using the Boltzmann relationship.

4.2 String method

The minimum-energy path (MEP) can be found on the landscape using the String method by defining the starting point, the end point, the step size, etc. As three data points are too few for demonstrating the String method, we use the landscape of all 1,000 volumes ('output_big.npy' and 'num_big.txt') instead to demonstrate its application.

python ../../ManifoldLandscape/string_method.py --landscape output_big.npy --number num_big.txt --range 160 --start -50 100 --end -80 -30 --stepsize 0.1 --interpolate cubic --kcenters 7

The input files of this step are 'output_big.npy' and 'num_big.txt', and the outputs are the energy landscape 'el.pdf' and the clustering centers 'centers_k.npy' along the MEP. In this example, we plot the energy landscape of substrate-bound 26S proteasome. Moreover, an MEP corresponding to substrate translocation initiation is also marked on the landscape (lemon line).

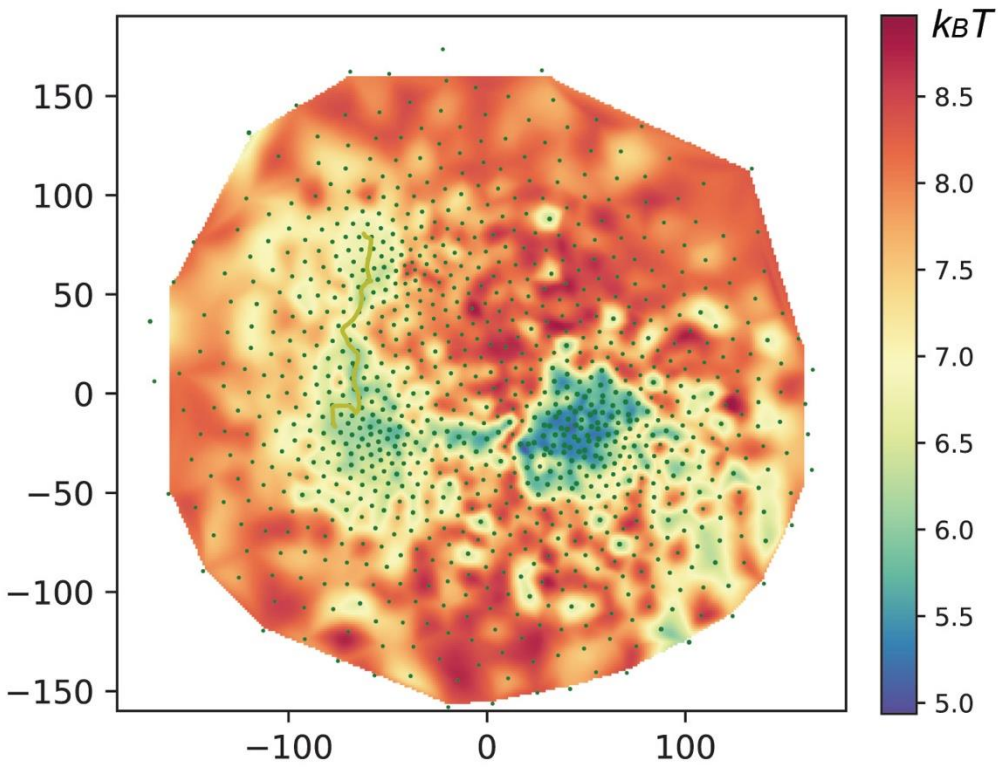

Fig. 2. Energy landscape of 26S proteasome during substrate translocation

5. Particle voting based on energy

5.1 3D classification via energy landscape

At this step, the clusters are prepared according to the centers and radii of the voting boundary on the energy landscape. Alternatively, we can just define the clusters manually if the landscape is simple enough, skipping minimum-energy path plotting.

cd ../ParticleVoting
python ../../ParticleVoting/clustering.py --points ../ManifoldLandscape/output_big.npy --centers ../ManifoldLandscape/centers_k.npy --radius 30

The input files of this step are '../ManifoldLandscape/output_big.npy' and '../ManifoldLandscape/centers_k.npy', and the outputs are 7 folders named by 'c' and class identifier. And each folder contains a file recording the data points of this 3D class.

5.2 Particle voting

For each 3D class, the corresponding star files are linked to the folder for particle voting respectively. Here also we use a small cluster 'c1_small' containing the previously mentioned three data points instead of 'c1' to

demonstrate the process of particle voting.

cd c1
. vote_prepare.sh c1_small

The inputs of this step are the 3D volumes' star files in the folder of '../Bootstrap/stars/'. So note that the 14th line of the script 'vote_prepare.sh' should also refer to this path. The outputs are the link files of these star files in the current directory.

Before voting on particles, the header of these particle star files are extracted.

python ../../../ParticleVoting/gethead.py --star u123_b1_c1.star

The input file of this step is any linked star file in the current directory such as 'u123_b1_c1.star', and the output file is 'head.star'.

Lastly, we need to run the voting script 'post_and_f.sh' to obtain the voted particles for high resolution refinement. Note that the voting threshold (M/2 is recommended) can be defined in the 2nd line, and enlarging this threshold may increase the classification accuracy and decrease the particle number of each class. The inputs of this step are all linked star files in the current directory and 'head.star', and the outputs are the files both before voting ('pre_vote.log' and 'pre_vote.star') and after voting ('post_vote.log' and 'post_and.star'). The star file 'post_and.star' is what we need.

. post_and_f.sh