GLOW: a workflow integrating Gaussian accelerated molecular dynamics and Deep Learning for free energy profiling

GLOW integrates Gaussian accelerated molecular dynamics (GaMD) and Deep Learning (DL) for free energy profiling of biomolecules. First, all-atom GaMD enhanced sampling simulations are performed on biomolecules of interest. Structural contact maps are then calculated from GaMD simulation frames and transformed into images for building DL models using convolutional neural network (CNN). Important structural contacts can be determined from DL models of saliency (attention) maps of the residue contact gradients, which allow for the identification of system reaction coordinates (RCs). Finally, free energy profiles of these RCs are calculated through energetic reweighting of GaMD simulations.

1. Gaussian accelerated molecular dynamics (GaMD)

Gaussian accelerated molecular dynamics (GaMD) is a robust computational method for simultaneous unconstrained enhanced sampling and free energy calculations of large biomolecules. GaMD works by adding a harmonic boost potential to smooth the potential energy surface when the system potential drops below a reference energy *E*:

$$\Delta V(r) = \begin{cases} \frac{1}{2}k\big(E - V(r)\big)^2, & V(r) < E \\ 0, & V(r) \geq E, \end{cases} \quad (1)$$

where *k* is the harmonic force constant. The two adjustable parameters *E* and *k* can be determined based on three enhanced sampling principles as described in previous studies. First, for any two arbitrary potential values $V_1(\vec{r})$ and $V_2(\vec{r})$ found on the original

energy surface, if $V_1(\vec{r}) < V_2(\vec{r})$, ΔV should be a monotonic function that does not change the relative order of the biased potential values; i.e., $V_1^*(\vec{r}) < V_2^*(\vec{r})$. Second, if $V_1(\vec{r}) < V_2(\vec{r})$, the potential difference observed on the smoothed energy surface should be smaller than that of the original, i.e., $V_2^*(\vec{r}) - V_1^*(\vec{r}) < V_2(\vec{r}) - V_1(\vec{r})$. The reference energy needs to be set in the following range:

$$V_{max} \leq E \leq V_{min} + \frac{1}{k}, \tag{2}$$

where V_{max} and V_{min} are the system minimum and maximum potential energies. To ensure that **equation (2)** is valid, k must satisfy: $k \leq \frac{1}{V_{max} - V_{min}}$. Let us define $k \equiv k_0 \frac{1}{V_{max} - V_{min}}$, then $0 \leq k_0 \leq 1$. Third, the standard deviation of ΔV needs to be small enough (i.e., narrow distribution) to ensure proper energetic reweighting: $\sigma_{\Delta V} = k(E - V_{avg})\sigma_V \leq \sigma_0$, where V_{avg} and σ_V are the average and standard deviation of the system potential energies, $\sigma_{\Delta V}$ is the standard deviation of ΔV with σ_0 as a user-specified upper limit (e.g., $10k_BT$) for proper reweighting. When E is set to the lower bound $E = V_{max}$, k_0 can be calculated as:

$$k_0 = \min(1.0, k_0') = \min\left(1.0, \frac{\sigma_0}{\sigma_V} \frac{V_{max} - V_{min}}{V_{max} - V_{avg}}\right), \tag{3}$$

Alternatively, when the threshold energy E is set to its upper bound $E \leq V_{min} + \frac{1}{k}$, k_0 is set to:

$$k_0 = k_0'' \equiv \left(1.0 - \frac{\sigma_0}{\sigma_V}\right) \frac{V_{max} - V_{min}}{V_{max} - V_{avg}}, \tag{4}$$

if k_0'' is found to be between 0 and 1. Otherwise, k_0 is calculated using **equation (3)**.

GaMD provides options to add only the total potential boost ΔV_P, only dihedral potential boost ΔV_D, or the dual potential boost (both ΔV_P and ΔV_D). The dual-boost GaMD

generally provides higher acceleration than the other two types of simulations. GaMD has been implemented in AMBER, NAMD, OpenMM, GENESIS and TINKER-HP, which could be applied for running the simulations.

2. Deep Learning (DL)

Deep learning (DL) is applied to analyze GaMD simulations of biomolecules. The residue contact map of each GaMD simulation frame is computed using Python packages MDTraj and contact map explorer. A contact definition of $\leq$4.5Å between any heavy atoms is used. The residue contacts maps are transformed into grayscale images of the same size for analysis by a 2D CNN. 80% of resulting images are used for training, and the rest are used for validation. The 2D CNN is built using the Keras module embedded in Python Tensorflow package. It consists of four convolutional layers of 3 x 3 kernel size, with 32, 32, 64 and 64 filters, respectively, followed by three fully connected (dense) layers, the first two of which include 512 and 128 filters with a dropout rate of 0.5 each. The final fully connected layer is the classification layer. “ReLu” activation is used for all layers in the 2D CNN, except the classification layer, where “softmax” activation is used. A maximum pooling layer of 2 x 2 kernel size is added after each convolutional layer. This model architecture has been tested to work for image-transformed structural contact maps of membrane proteins with ~270 – 290 residues and classification of ~10 different systems. For more complicated classifications, additional convolutional and/or dense layers may be added for DL. Finally, important residue contacts are determined by DL and backpropagation by vanilla gradient-based pixel attribution using residue contact maps of biomolecules.

3. Free energy profiling

Reaction coordinates associated with important structural contacts identified from DL are selected to compute free energy profiles by energetically reweighting the GaMD simulations. The probability distribution of selected RCs can be calculated from simulations as $p^*(A)$. Given the boost potential $\Delta V(r)$ of each frame in GaMD simulations, $p^*(A)$ can be reweighted to recover the canonical ensemble distribution, $p(A)$, as:

$$p(A_j) = p^*(A_j)\frac{\langle e^{\beta \Delta V(r)}\rangle_j}{\sum_{i=1}^{M}\langle p^*(A_i)e^{\beta \Delta V(r)}\rangle_i}, \qquad j = 1, \dots, M \tag{5}$$

where *M* is the number of bins, $\beta = k_B T$ and $\langle e^{\beta \Delta V(r)}\rangle_j$ is the ensemble-averaged Boltzmann factor of $\Delta V(r)$ for simulation frames found in the j^{th} bin. The ensemble-averaged reweighting factor can be approximated using cumulant expansion:

$$\langle e^{\beta \Delta V(r)}\rangle = exp\left\{\sum_{k=1}^{\infty}\frac{\beta^k}{k!}C_k\right\}, \tag{6}$$

where the first two cumulants are given by:

$$\begin{aligned} C_1 &= \langle \Delta V\rangle, \\ C_2 &= \langle \Delta V^2\rangle - \langle \Delta V\rangle^2 = \sigma_v^2. \end{aligned} \tag{7}$$

The boost potential obtained from GaMD simulations usually shows near-Gaussian distribution. Cumulant expansion to the second order thus provides a good approximation for computing the reweighting factor. The reweighted free energy $F(A) = -k_B T \ln p(A)$ is calculated as:

$$F(A) = F^*(A) - \sum_{k=1}^{2}\frac{\beta^k}{k!}C_k + F_c, \tag{8}$$

where $F^*(A) = -k_B T \ln p^*(A)$ is the modified free energy obtained from GaMD simulation and F_c is a constant.

A toolkit of Python scripts "PyReweighting" has been developed to facilitate reweighting analysis of GaMD simulations. PyReweighting implements a list of commonly used reweighting methods, including (1) exponential average that reweights trajectory frames by the Boltzmann factor of the boost potential and then calculates the ensemble average for each bin, (2) Maclaurin series expansion that approximates the exponential Boltzmann factor, and (3) cumulant expansion that expresses the reweighting factor as summation of boost potential cumulants. Notably, Maclaurin series expansion is equivalent to cumulant expansion on the first order. Cumulant expansion to the second order ("Gaussian approximation") normally provides the most accurate reweighting results. The PyReweighting scripts and tutorial can be downloaded at http://miaolab.org/PyReweighting.

4. GLOW usage

In the current implementation of GLOW, users should have set up the simulation systems and performed energy minimization, equilibration with the constant number, volume and temperature (NVT) ensemble, equilibration with the constant number, pressure and temperature (NPT) ensemble and a short conventional MD (cMD).

The system parameter files (**prm, prmtop** or **parm7**) and the cMD resulting restart files (**rst** or **rst7**) are the only required inputs for GLOW. All the flags and variables to run GLOW are defined in the **GLOW.in** input file. It is noteworthy that current GLOW scripts support only GaMD simulations of membrane proteins using AMBER and DL analysis of structural contact maps. Future scripts will be added to support GaMD simulations of globular proteins, simulations using other software packages (e.g., NAMD, OpenMM,

GENESIS, TINKER, etc.), and DL analysis of atomic coordinates. Furthermore, current GLOW implementation still requires users to plot free energy profiles using outputs of GaMD reweighting calculations, which will be also automated in the future.

The source codes of GLOW, along with testing "EXAMPLE" folder, are provided at http://miaolab.org/GLOW. The script **run-GLOW.sh** is used to run GLOW in full or parts, while other shell scripts will generate all the necessary files and scripts for running GLOW.

To run GLOW (in full or in parts), type the following command in the terminal: ***(nohup) ./run-GLOW.sh &*** AND **ALWAYS use *CTRL A + CTRL D*** to log off the terminal and not disrupt the running process.

The following parameters in the **GLOW.in** input file are to be used to define which parts of GLOW to be run:

run_GaMD	Set to ***1*** to run GaMD simulations or ***0*** to skip GaMD simulations.
run_GaMD_analysis	Set to ***1*** to perform GaMD simulation analysis or ***0*** to skip GaMD simulation analysis.
run_DL_prep	Set to ***1*** to calculate residue contact maps of GaMD simulation frames and transform them into images or ***0*** to skip calculations and transformations of residue contact maps.
run_2D_CNN	Set to ***1*** to perform DL analysis of image-transformed residue contact maps or ***0*** to skip DL analysis of image-transformed residue contact maps.

run_DL_analysis	Set to ***1*** to plot DL metrics, confusion matrix and calculate important residue contacts from residue contact maps or ***0*** to skip.
run_2D_FEPs	Set to ***1*** to print out the instructions on how to plot 2D free energy landscapes or ***0*** to skip.

Each part of GLOW from above requires the specifications of a number of variables to run properly. First, the number of simulation systems is defined with the variable:

nb_systems	Number of biomolecule systems. Must be a positive integer (2, 3, 4 …). GLOW has been tested to work for ~10 different systems.

In the example folder, only two systems are included for demonstration of GLOW. In case users have more than two systems, the new variables must be in **identical** formats to the existing ones (***same_text_$i*****, with $i = 1, 2, 3, …**)

For Part 1: GaMD, the following variables must be defined:

AMBER	Path to the folder where the ***amber.sh*** or ***amber.csh*** file is located.
workfolder_$i	Path to the folder where GaMD simulation input (parameter and restart files) and output files of system $i ($i = 1, 2, 3…) are located.
parm_sys_$i	Name of system $i ($i = 1, 2, 3…) parameter file that contains the whole complex. This ***parm7*** or ***prmtop*** file should be available prior to start GLOW. *Default: system-$i-sys.parm7*

rst_sys_$i	Name of system $i ($i = 1, 2, 3…) cMD restart file. This ***rst*** or ***rst7*** file should be available prior to start GLOW. *Default: step7_10.rst7*
pdb_sys_$i	Name of system $i ($i = 1, 2, 3…) pdb file that contains the whole complex. This ***pdb*** file is generated by GLOW. *Default: system-$i-sys.pdb*
nb_prot_$i	Number of protein and ligand atoms contained in system $i ($i = 1, 2, 3…).
parm_prot_$i	Name of system $i ($i = 1, 2, 3…) parameter file that contains only protein and ligand residues. This ***parm7*** or ***prmtop*** file is generated by GLOW. *Default: system-$i-pro.parm7*
pdb_prot_$i	Name of system $i ($i = 1, 2, 3…) pdb file that contains only protein and ligand residues. This ***pdb*** file is generated by GLOW. *Default: system-$i-pro.pdb*
res_idx_$i	Range of residue index to keep for calculations of residue contact maps of system $i ($i = 1, 2, 3…). The final numbers of residues should be identical across systems.
parm_cmap_$i	Name of system $i ($i = 1, 2, 3…) parameter file that contains only residues to generate residue contact maps. This ***parm7*** or ***prmtop*** file is generated by GLOW. *Default: system-$i-cmap.parm7*

pdb_cmap_$i	Name of system $i ($i = 1, 2, 3…) pdb file that contains only residues to generate residue contact maps. This ***pdb*** file is generated by GLOW. *Default: system-$i-cmap.pdb*
traj_cmap_$i	Name of system $i ($i = 1, 2, 3…) trajectory file that contains only residues to generate residue contact maps. This ***nc*** file is generated by GLOW. *Default: system-$i-cmap.nc*
total_prod_steps	Number of GaMD production simulation steps for all systems. The simulation length will be equal to *(this variable x 0.002) ps*. *Default: 500,000,000*
stride	Number of simulation frames to be skipped in calculations of structural contact maps, e.g., *10 (default)* will have residue contact maps generated for every 10th simulation frame.

For Part 2: DL, the following variables must be defined:

cuDNN_lib	Path to the folder where ***lib*** folder of CUDA is located to run DL on CUDA GPUs. *Default: ~/ cuda-11.0/targets/x86_64-linux/lib/*
dl_dir	Path to the folder where DL-related files and scripts are deposited and run.
sys_fold_$i	Name of the folder where generated residue contact maps of system $i ($i = 1, 2, 3…) are deposited. This folder is made by GLOW.
sys_img_$i	Pivotal parts of the names of image-transformed residue contact maps from GaMD simulation frames of system $i ($i = 1, 2, 3…).

	The resulting names have the form of *pivot-0.jpg, …, pivot-99999.jpg*
nb_residues	Number of residues in the residue contact maps across the systems. GLOW has been tested for membrane proteins with ~270 – 290 residues.
image_index	Index of image-transformed residue contact map used to determine important residue contacts (e.g., the contact map *pivot-(image_index).jpg* will be used). This must be from the validation datasets and towards the end of GaMD production simulations. *Default: 99999*
gradient_cutoff	Gradient cutoff to determine important residue contacts (pixels) from residue contact maps. *Default: 0.4* but can be any between 0 and 1.

5. Further information

DL: Required Python packages are recommended to be installed in Anaconda3 environment of Python3.7+. Detailed instructions to install Anaconda3 on Linux can be found at https://docs.anaconda.com/anaconda/install/linux/. Below is an overview:

1. Download the installation *bash* file of Anaconda3 (*Anaconda3-2021.05-Linux-x86_64.sh or later)* from:

https://www.anaconda.com/products/individual

2. To install Anaconda3, run the *bash* file by the following command line:

./Anaconda3-2021.05-Linux-x86_64.sh

3. After the installation, close and reopen the terminal. If you see *(base)* at the start of terminal lines, then the installation has been successful.

4. Create a python3.7+ environment by: ***conda create -n py3 python=3.8***

5. Activate the created python3.7+ environment by: ***conda activate py3***

6. To deactivate the environment, use the command: ***conda deactivate***

For installation of necessary Python packages, run **install-PyPackages.sh** by the following command line:

./install-PyPackages.sh

The Python packages for DL include:

- MDTraj: https://www.mdtraj.org/1.9.5/index.html
- Contact map explorer: https://contact-map.readthedocs.io/en/latest/
- numpy: https://numpy.org/
- Python Image Library (PIL): https://pillow.readthedocs.io/en/stable/
- Tensorflow-GPU (version: 2.4.0+): https://www.tensorflow.org/install/gpu
- pickle: https://docs.python.org/3/library/pickle.html
- matplotlib (and pylab): https://matplotlib.org/
- scikit-learn: https://scikit-learn.org/stable/
- tf-keras-vis: https://keisen.github.io/tf-keras-vis-docs/

Reference

Do HN, Wang J, Bhattarai A, and Miao Y. (2021) GLOW: A workflow integrating Gaussian accelerated molecular dynamics and Deep Learning for free energy profiling. *In Prep.*