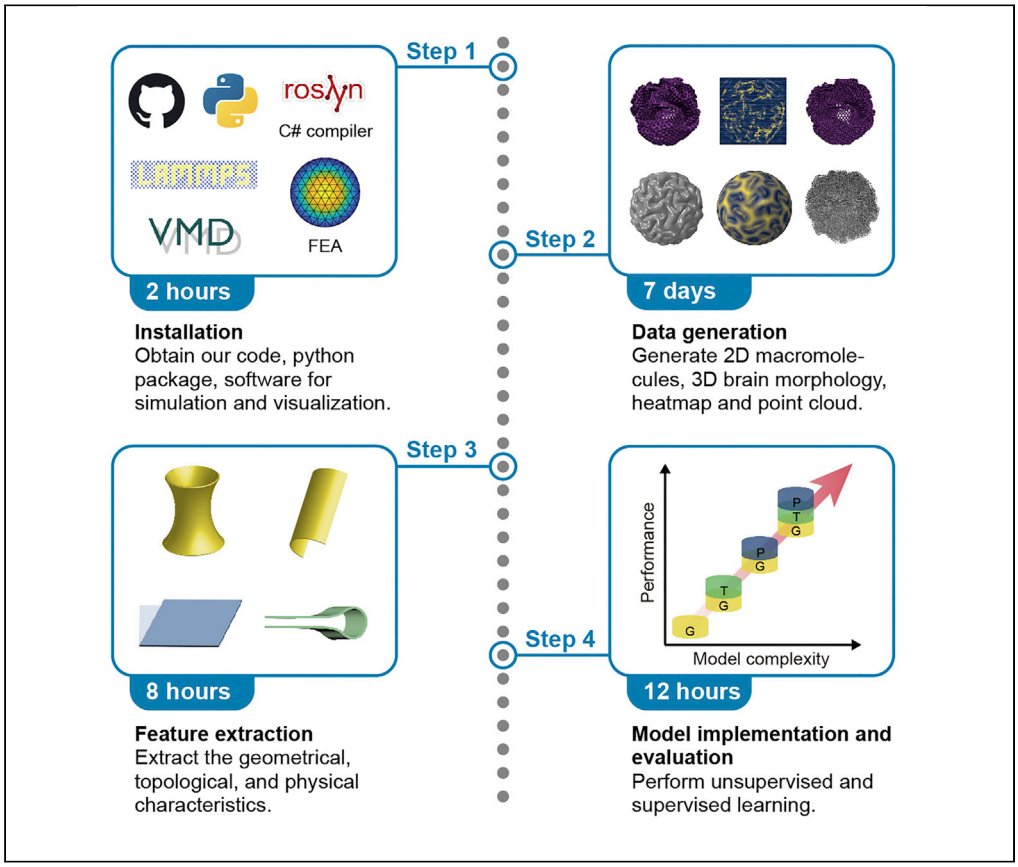
STAR Protocols

CellPress
OPEN ACCESS

Protocol

A statistical learning protocol to resolve the morphological complexity of two-dimensional macromolecules



Unraveling the morphological complexity of two-dimensional macromolecules allows researchers to design and fabricate high-performance, multifunctional materials. Here, we present a protocol based on statistical learning to resolve morphological complexity utilizing geometrical, topological, and physical features extracted from the strain energy heatmap and structural point cloud. We detail steps for software installation and data generation. We further describe model implementation and evaluation via unsupervised and supervised learning and discuss a theoretical description of morphological complexity including topological features.

Publisher's note: Undertaking any experimental protocol requires adherence to local institutional guidelines for laboratory safety and ethics.

Yingjie Zhao,
Zhiping Xu

zhaoyj21@mails.tsinghua.edu.cn (Y.Z.)
xuzp@tsinghua.edu.cn (Z.X.)

Highlights

A statistical learning protocol to resolve morphological complexity of macromolecules

Detailed steps to extract geometrical, topological, and physical features

A framework combining unsupervised and supervised learning for model implementation

Codes are provided for data generation, feature extraction, and model implementation

Zhao & Xu, STAR Protocols 3, 101767
December 16, 2022
https://doi.org/10.1016/j.xpro.2022.101767

STAR Protocols

CellPress
OPEN ACCESS

Protocol

A statistical learning protocol to resolve the morphological complexity of two-dimensional macromolecules

Yingjie Zhao[1,2,*] and Zhiping Xu[1,3,*]

[1]Applied Mechanics Laboratory, Department of Engineering Mechanics and Center for Nano and Micro Mechanics, Tsinghua University, Beijing 100084, China

[2]Technical contact

[3]Lead contact

*Correspondence: zhaoyj21@mails.tsinghua.edu.cn (Y.Z.), xuzp@tsinghua.edu.cn (Z.X.)
https://doi.org/10.1016/j.xpro.2022.101767

SUMMARY

Unraveling the morphological complexity of two-dimensional macromolecules allows researchers to design and fabricate high-performance, multifunctional materials. Here, we present a protocol based on statistical learning to resolve morphological complexity utilizing geometrical, topological, and physical features extracted from the strain energy heatmap and structural point cloud. We detail steps for software installation and data generation. We further describe model implementation and evaluation via unsupervised and supervised learning and discuss a theoretical description of morphological complexity including topological features.

For complete details on the use and execution of this protocol, please refer to Zhao et al. (2022).

BEFORE YOU BEGIN

A statistical learning model combining unsupervised and supervised learning which embeds the geometrical, topological, and physical features was proposed to unravel the morphological complexity of 2D macromolecules (Zhao et al., 2022) and extended to 3D model brain (Razavi et al., 2015). This protocol provides detailed guidance for generating the data, extracting the features, and implementing the model. Before you begin, this section describes the preparations including hardware condition, software installation, the download of our code, the format of the dataset, and the environment configuration required to run the code.

Hardware condition

Timing: 10 min

1. We deploy the protocol on CentOS Linux release 7.6.1810 (on a CPU cluster), CentOS Linux release 7.4.1708 (on a GPU cluster), and Windows 10 (on a laptop). The CPU type is Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40 GHz and the GPU type is NVIDIA GeForce GTX 2080Ti.

Note: The code and software are functional on all operating systems such as Linux, Mac OS, and Windows, and you only need to install the software of the specified operating system version. The estimated cost time of the following steps is based on the above hardware configuration, but the actual time spent depends on your hardware condition, such as the

parallelization would reduce the required time and the lower hardware configuration would take more time.

Software installation and code download

Timing: 1.5 h

2. The Python code used to implement the models needs to run in a configured environment. Anaconda is a helpful platform to run Python on different operating systems and is convenient for package and environment management. Install Anaconda according to your operating system type from this link: https://www.anaconda.com/products/distribution and the version of Python is recommended as 3.7.
3. Install required libraries and packages on the created virtual environment, and the version and source could refer to the key resources table. It can be realized in the graphical user interface (GUI) or command prompt console.
   a. In the GUI, launch Anaconda Navigator, choose the 'Environments' module, click the 'Create', input the name of your virtual environment, and select the corresponding Python version. Then you could search and install the required packages on your created environment.
   b. In the command prompt console, you should create and activate the virtual environment and then install the needed packages using the following command:

```
>conda create -n your_environment_name python=3.7

>conda activate your_environment_name

>pip install tensorflow==2.3

>pip install keras==2.3.1 >pip install numpy==1.18.5 >pip install opencv-python==4.5.1.48 >pip install scikit-learn==0.24.1
```

4. We use the C# script to generate input files for the simulation in batches and recommend the open source compiler Roslyn to run the script following this link: https://github.com/dotnet/roslyn.
5. We utilize molecular simulations to generate the 2D macromolecules morphology based on a coarse-grained model (refer to Zhao et al., 2022 for more details). Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS) is an open-source molecular dynamics program that supports parallel communication. Download LAMMPS according to your operating system type and serial or parallel capabilities from this link: https://www.lammps.org/download.html. An input command script and a data file are generally needed to run the simulation.
   a. Specify the path of executable file 'lmp_serial' (for serial version) or 'lmp_mpi' (for parallel version) to the environment variable.
   b. Enter the folder where your input script is located and type the following command to run a simulation. For more detailed guidance, please refer to https://docs.lammps.org/Manual.html.

```
>cd your_work_directory

>lmp_serial -in in.file

>mpirun -np 4 lmp_mpi -in in.file
```

6. We adopt the finite element analysis (FEA) to generate 3D brain morphology based on a thermo-mechanical continuum model (refer to Razavi et al., 2015 for more details). The model is not a real

physical model of the brain but could generate characteristic sulci and gyri to simulate the brain morphology and analyze the structural complexity. Enter the working directory and type the following command to run a simulation.

```
>cd your_work_directory

>abaqus job=your_job_name cpus=6 int
```

Note: The large deformation and contact features of brain morphology lead to the geometrical, material, and contact nonlinearities. ABAQUS is a software suite for nonlinear FEA, however, it is not an open source software, and you could contact sales from this link: https://www.3ds.com/products-services/simulia/products/abaqus. Code_aster as an optional open source software of FEA which could handle the nonlinear and thermomechanical problems is also recommended, following this link: https://www.code-aster.org for download and documentation.

7. Download Visual molecular dynamics (VMD) according to your operating system type from this link: https://www.ks.uiuc.edu/Development/Download/download.cgi?PackageName=VMD. We utilize it to visualize atom energy, the structure of point cloud, and calculate geometrical features of morphology which would be demonstrated later.

Note: The generated morphological data need to be visualized and analyzed properly. VMD is an open source that visualizes the results of molecular simulation and supports the Tcl script for batch operations.

8. Download the code and scripts from the GitHub repository: https://github.com/zhaoyj21/morphology_complexity. These files include the source of data generation, feature extraction, and model implementation, which would be exhibited step by step later.

Dataset and preparation of running code

Timing: 20 min

9. We generate a total of 2484 samples and resulting in three datasets including extracted features, heatmap, and point cloud. The extracted features dataset has a format of the array with the size of 2484*6, containing characteristics of three principal features, the radius of gyration (R_g), solvent accessible surface area (SASA), and localization factor (L_F). The heatmap dataset contains 2484 images of the strain energy map. The point cloud dataset includes 2484 'txt' files storing the coordinates of atoms. The process of data generation would be demonstrated in detail later.
10. Before you run the code, please activate the virtual environment which has configured the required packages first. Then enter the working directory and launch Jupyter notebook to run the 'ipynb' file or directly run the 'py' file.

```
>conda activate your_environment_name

>cd your_work_directory

>python3 name_of_work.py
```

KEY RESOURCES TABLE

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Deposited data		
Source code	This paper	https://github.com/zhaoyj21/morphology_complexity
Software and algorithms		
Anaconda 4.11.0	Anaconda	RRID: SCR_018317; https://www.anaconda.com/products/distribution
Python 3.6.8	Python Software Foundation	RRID: SCR_008394; https://www.python.org/downloads/
TensorFlow 2.3	TensorFlow	RRID: SCR_016345; https://www.tensorflow.org/install
Keras 2.3.1	Keras	https://github.com/keras-team/keras
CUDA 10.1	NVIDIA	https://developer.nvidia.com/cuda-10.1-download-archive-base
cuDNN 7.6.5	NVIDIA	https://developer.nvidia.com/rdp/cudnn-archive
NumPy 1.18.5	NumPy	RRID: SCR_008633; https://numpy.org/install/
OpenCV-python 4.5.1.48	OpenCV	RRID: SCR_015526; https://github.com/opencv/opencv
Scikit-learn 0.24.1	scikit-learn	RRID: SCR_002577; https://scikit-learn.org/stable/install.html
Roslyn	Microsoft	https://github.com/dotnet/roslyn
LAMMPS	Plimpton (1995)	https://www.lammps.org/download.html
ABAQUS	3DEXPERIENCE	https://www.3ds.com/products-services/simulia/products/abaqus
VMD	Humphrey et al. (1996)	https://www.ks.uiuc.edu/Development/Download/download.cgi?PackageName=VMD
Other		
Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40 GHz	Intel	RRID: SCR_016973
NVIDIA GeForce GTX 2080Ti	NVIDIA	N/A
CentOS Linux release 7.6.1810	CentOS	N/A
CentOS Linux release 7.4.1708	CentOS	N/A
Windows 10	Microsoft	N/A

STEP-BY-STEP METHOD DETAILS

In this section, detailed guidance of the protocol based on statistical learning to unravel the morphological complexity is described step-by-step, including data generation, feature extraction, and model implementation and evaluation as exhibited in Figure 1.

Data generation

Timing: 7 days

The following steps in this part exhibit the process of data generation including the morphology of 2D macromolecules and 3D brain as demonstrated in Figure 2.

1. To generate data, obtain the input files in batches via changing simulation parameters for 2D macromolecules. Troubleshooting 1.
   a. Open the 'Program.cs' file in the './AutoFilesSP' folder.
   b. Custom the parameters.
   c. Specify the path.

   Note: The path './AutoFilesSP' means the 'AutoFilesSP' folder in the current directory where you download from GitHub (https://github.com/zhaoyj21/morphology_complexity) to your local computer. You can change the simulation parameters and the save path of generated files. However, the custom parameters should vary in an appropriate range which can refer to our default values. In addition, the path of the original input files used for adjusting parameters should be specified accurately.

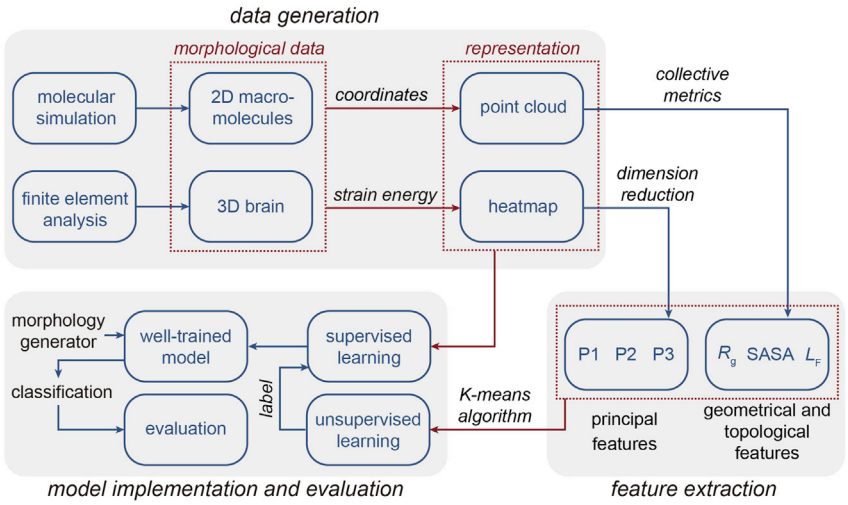


Figure 1. The guideline of the protocol based on statistical learning to unravel the morphological complexity of 2D macromolecules and 3D brain

d. Run the 'Program.cs' file using the C# compiler Roslyn after specifying the path and parameters.

Optional: You can choose any C# compiler or custom script in any programming language to achieve the obtainment of input files in batches.

Note: The input folders 'sphere' would be generated in batches and each folder should contain the 'in.sphere' and 'l100.data' files which are used for the control script and the data file in LAMMPS package. The 'in. sphere' with different parameters is generated by the 'Program.cs' file based on the original input script. The input script describes the operation command in LAMMPS, and the meaning of the command could refer to the manual on the official website: https://docs.lammps.org/Manual.html. The data file could be generated by software package such as Materials Studio or custom script according to structural characteristics. Here, we provide the original input script and data files for use in the './origin' folder.

2. Run the input files in batches utilizing script and LAMMPS package. Troubleshooting 2 and 3.
   a. Enter the input folders and run the bash script:

```
>cd ./sphere

>./batch-sp.bash
```

Note: We run the script and perform the calculation on the Explorer 100 cluster system of Tsinghua National Laboratory for Information Science and Technology. You can also perform these on your desktop or laptop, but the process may take much time and computational cost.

Pause point: A morphological sample would take 30 min in our hardware configuration, and a batch task with 120 samples would take 2.5 days.

b. Check the desired output files.

Note: The 'test.xyz' file contains the coordinates of atoms at different time steps and the 'relax.dump' file includes the coordinates and energy of atoms of the equilibrium structure after relaxation.

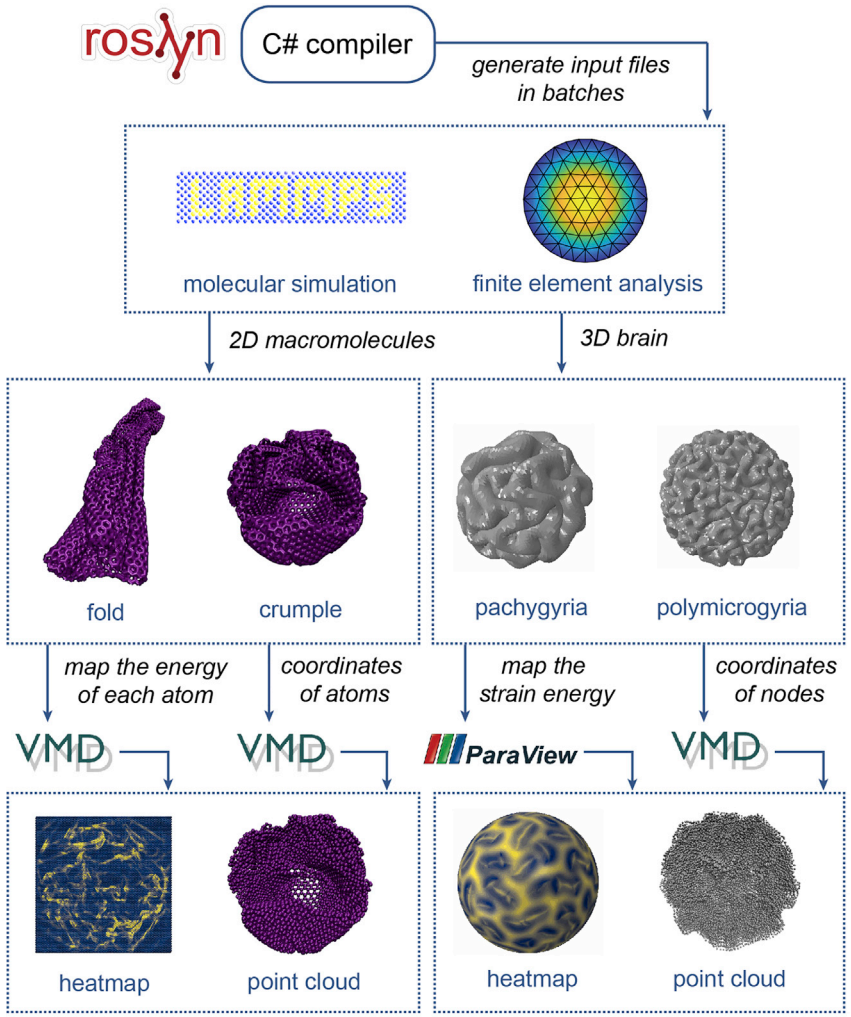


Figure 2. Flowchart of data generation, including 2D macromolecules, 3D model brain morphology as well as corresponding strain energy map, and point cloud

3. After finishing the simulation, visualize and analyze the morphological data. Troubleshooting 2 and 3.

   Note: The heatmap of strain energy contains the deformation information of conformation. We generate the heatmap of 2D macromolecules via map the energy of atoms to the initial configuration.

   a. Enter the './ridge' folder and compile the source code as shared library 'libread.so'.

```
>cd ./ridge

>gcc -shared -fPIC ./src/*c -o libread.so
```

   Note: The source code is written in C programming language, and compiling it as the shared library is convenient to call by Python.

b. Enter the './sphere' folder and run the 'cpvel.bash' file.

```
>cd ./sphere

>./cpvel.bash
```

Note: You should modify the specific path in the bash script and make sure to find the required files when running the code. The energy data of each sample is saved in the 'Edis.vel' file of the './sphere_vel' folder.

c. Open the VMD software and click the 'Extensions' module, select the 'TK console' function.
d. Click the 'File' module and select the 'Load File' function.
e. Choose the 'VMDdrawing.tcl' script to accomplish the generation of the heatmap for each sample.

Note: The format of the heatmap is 'png' and all images are saved in the './2Dimage' folder. Similarly, please check the specific path in the bash script and you can custom the save path of the output.

4. Obtain the point could of 2D macromolecules morphology. Troubleshooting 2 and 3.

Note: Besides the deformation information, the geometrical shape is also key for morphology characterization. The point cloud containing the spatial coordinates of atoms could properly represent the geometrical features.

a. Enter the './sphere' folder and run the 'cpdump.bash' script to copy the 'relax.dump' file in each sample to the './DUMP' folder and number them in order.

```
>cd ./sphere

>./cpdump.bash
```

b. Open the 'Program.cs' file in the './DUMPtoTXT-sphere' folders.
c. Run the 'Program.cs' to convert the 'relax.dump' files to 'txt' files containing point cloud information and save them to the './TXT' folder.
d. You can also run the 'Program.cs' through following command line:

```
>csc Program.cs

>Program.exe
```

Note: The 'txt' files are saved by reading the 'relax.dump' files and cutting the desired coordinates information utilizing the script file 'Program.cs'. You can custom the parameter of the saving path in the 'Program.cs' file via modifying the value of string 'aimPath'.

e. Enter the './sphere' folder and run the 'cpxyz.bash' script to copy the 'test.xyz' file in each sample to the './XYZ_pre' folder and number them in order.

```
>cd ./sphere

>./cpxyz.bash
```

Note: The point cloud data could also be saved as the 'xyz' format, which is convenient for the feature extraction later using the algorithm embedded in VMD.

f. Open the 'Program.cs' file in the './XYZ_cut' folder and run the code.

Note: Please check the path specification in all scripts. The code would cut the morphology of the last frame to the './XYZ' folder which is used to extract features later.

5. Generate data of 3D brain morphology. Troubleshooting 2 and 3.

Note: Beyond 2D macromolecules, the framework of unraveling morphological complexity could be extended to the 3D model brain. We also generate the 3D brain morphology, and corresponding energy heatmap and point cloud.

a. Enter the './brain_thin1' folder and summit the 'brain_thin1.inp' file in ABAQUS.

```
>cd ./brain_thin1

>abaqus job=brain_thin1 cpus=6 int
```

Note: The desired output is the 'brain_thin1.odb' file containing the nodal coordinates of deformed configuration and strain energy.

b. Generate the heatmap of strain energy.
   i. Launching the ABAQUS CAE, click the 'Results' module, and Choose the 'Output Databases' function.
   ii. Select the result file 'brain_thin1.odb'.
   iii. Clicking the 'Plot Contours on Undeformed Shape' function.

Note: The obtained heatmap of 3D brain morphology is exhibited in Figure 2. You can also custom script to acquire the heatmap via the open source visualization software package ParaView utilizing the strain energy and initial configuration.

c. Extract the point cloud of 3D brain morphology.
   i. Launch the ABAQUS CAE, and click the 'File' module.
   ii. Select 'Import', 'Part', and choose the result file 'brain_thin1.odb'.
   iii. Check the 'Import deformed configuration' option, and select the conformation of the specific time step.
   iv. Write this morphology into the 'thin1_step7.inp' file.

Note: The 'inp' file could be written through the process of assembly, creating jobs, and writing input.

d. Change the suffix of 'thin1_step7.inp' to 'thin1_step7.txt'.
e. Launch Anaconda, click Jupyter notebook, and run the 'abq_convert_vmd.ipynb' file.

Note: We could obtain the 'deform_thin1_step7.xyz' and 'deform_thin1_step7.txt' as point cloud data used to extract structural features and morphological classification.

Feature extraction

Timing: 8 h

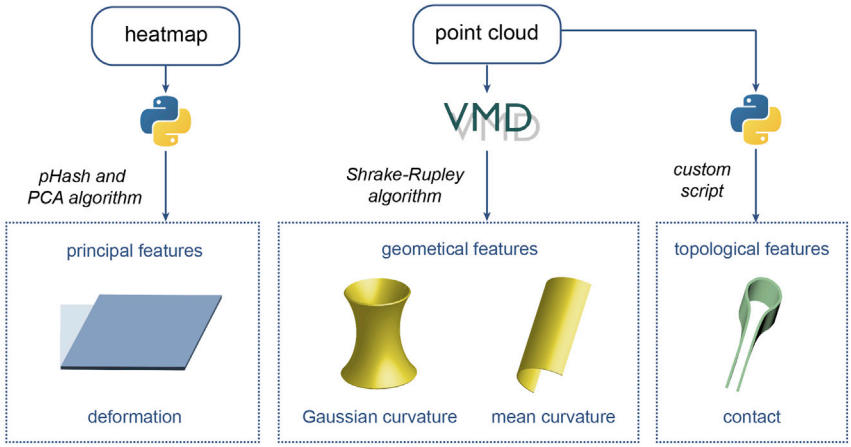


Figure 3. Schematic diagram of feature extraction, containing the principal characteristics from the heatmap, the geometrical and topological characteristics from the point cloud

After finishing the data generation process including the heatmap and the point cloud of the morphology, feature extraction is key to quantifying and analyzing the morphological data. The following steps in this section describe the process of characteristic extraction as shown in Figure 3, including principal feature of the heatmap, the geometrical radius of gyration (R_g) and solvent accessible surface area (SASA), and topological localization factor (L_F). The combination of physical, geometrical, and topological features could provide a reasonable characterization of the morphology.

6. Run the following code to implement the extraction of principal feature. Troubleshooting 3 and 4.

   Note: The deformation information is embedded in the heatmap. We utilize the perceptual Hash (pHash) algorithm (Monga et al., 2006) and principal component analysis (PCA) (Wold et al., 1987) to extract the principal feature of the heatmap, since pHash algorithm could distinguish the similarity of the heatmap and PCA is a commonly used linear dimension reduction technique. The combination of pHash and PCA achieves the nonlinear dimension reduction and feature extraction of the heatmap.

   a. Open Anaconda, select the configured virtual environment and launch Jupyter notebook.

   Note: Before running the code, ensure to activate the required virtual environment.

   b. Run the 'imagecluster_bin.ipynb' file and save all heatmap images to a binary file 'data_batch_1'.
   c. Run the 'pHash_and_PCA.ipynb' file and obtain the principal feature in 'p1.txt', 'p2.txt', and 'p3.txt' files.

7. Extract geometrical features utilizing VMD. Troubleshooting 1 and 3.

   Note: The geometrical curvature and shape features are represented by the point cloud. We use the algorithm embedded in VMD to extract the R_g and SASA, since R_g measures the shrinkage and SASA characterizes the exposure, and the combination of them could provide a proper geometrical representation of the morphology.

   a. Open the VMD software and click the 'Extensions' module.
   b. Select the 'TK console' function, click the 'File' module.
   c. Select the 'Load File' function.

d. Choose the 'VMDrgyr.tcl' to extract the R_g of each morphological sample.
e. Choose the 'VMDsasa.tcl' script to calculate the SASA of sample (Shrake and Rupley, 1973).

Note: The extracted features are saved in the 'rgyr.txt' and 'sasa.txt' files, respectively. The solvent probe radius is a key parameter in 'VMDsasa.tcl' script and you could custom it according to the structural size characteristics of your interesting problem.

8. Enter the './LF-sp' folder and run the 'LF-sp.py' file. Troubleshooting 1, 3, and 4.

```
>cd ./LF-sp

>python3 LF-sp.py
```

Note: The topological contact is the key feature that determines the molecular processes of adsorption and transport of morphology, therefore, we define L_F to represent the localization degree of surface contact based on the point cloud of the deformed and initial configuration (refer to Zhao et al., 2022 for more details). The cutoff of contact is a key parameter in the 'LF-sp.py' script and you could custom it according to the morphological characteristics of your problem. The topological contact characteristics for each sample are saved in the 'LF-sp.txt' file.

9. Extract key features of 3D brain morphology. Troubleshooting 1, 3, and 4.
   a. Open the 'deform_thin1_step7.xyz' file in VMD.
   b. Click the 'Extensions' module.
   c. Select the 'TK console' function, and type following command to calculate the R_g and SASA, respectively.

```
>measure rgyr [atomselect top all]

>measure sasa 5 [atomselect top all]
```

   d. Enter the './thin1_step7' folder and run the 'LF-brain.py' file.

```
>cd ./ thin1_step7

>python3 LF-brain.py
```

Note: The values of L_F would be printed on the screen. The morphological distribution of the 3D brain in the space of extracted features is exhibited in Figure 4. Adding the topological feature distinguishes the conformation better (Figure 4B) than the space containing only geometrical information (Figure 4A).

Model implementation

Timing: 12 h

The morphology could be distinguished in the space of extracted features based on an unsupervised learning model. The obtained labels from unsupervised learning could be applied to subsequent supervised learning. The well-trained statistic learning model combining unsupervised and supervised

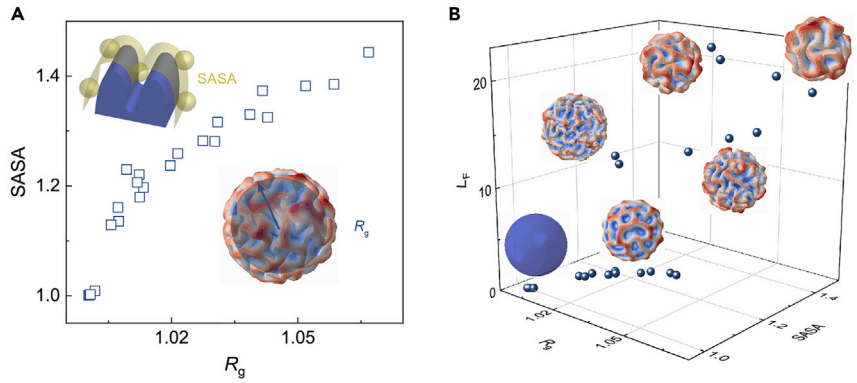


Figure 4. The morphological distribution of the 3D brain in the space of extracted features

(A) The characteristic space only contains the geometrical information.

(B) The space includes both geometrical and topological features.

learning would possess the ability of morphological complexity identification. The following steps in this section display the way to implement the statistic learning model as illustrated in Figure 5.

10. Cluster the morphological phases. Troubleshooting 3 and 4.

Note: We adapt the K-means algorithm to perform unsupervised learning based on the well-extracted features since this algorithm could cluster the morphology properly with similar characteristics (Hartigan and Wong, 1979).

a. Combine the 'rgyr.txt' and 'sasa.txt' into 'dataset_G.txt'.
b. Open Anaconda, select the configured virtual environment, and launch Jupyter notebook.
c. Run the 'K-means.ipynb' file.

Note: You could obtain the results 'label_G.txt' file for subsequent supervised learning. This label only contains the geometrical information of morphology.

d. Similarly, merge the 'rgyr.txt', 'sasa.txt', and 'LF.txt' into 'dataset_G+T.txt', and run the 'K-means.ipynb' file.

Note: The 'label_G+T.txt' containing both the geometrical and topological characteristics could be acquired. You can also combine the 'p1.txt', 'p2.txt', and 'p3.txt' and obtain the label based on physical deformation energy characterized by heatmap following the same process.

⚠ CRITICAL: The generated labels in this step contain different morphological information, inputting them to subsequent supervised learning could acquire different models (Figure 5).

11. Implement the supervised learning based on the point cloud dataset and the label obtained from unsupervised clustering. Troubleshooting 1, 3, 4, and 5.

Note: The point cloud contains the geometrical information. Using different labels could produce different models (Zhao et al., 2022). Here, we utilize the G+T model for the exhibition of process.

a. Convert the point cloud dataset './TXT' folder to the format needed by the PointNet algorithm according to the label file 'label_G+T.txt' (Qi et al., 2017).

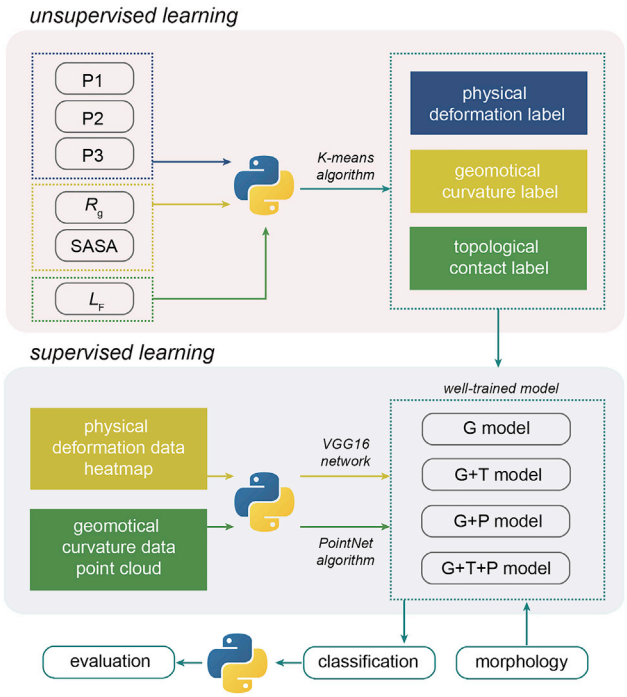


Figure 5. The framework of the statistic learning model combined unsupervised and supervised learning, which embeds the geometrical, topological, and physical features

i. Open the 'label_G+T.txt' in Excel and sort the sample number by the label, then write the bash script files 'cptxt-crumple.bash', 'cptxt-fold.bash', 'cptxt-interphase.bash', and 'cptxt-quasi-flat.bash'.
ii. Run the above four scripts.

Note: You would obtain the './pointnet.pytorch.3D/owndata' folder containing four subfolders in which includes the point cloud data corresponding to specific classification.

iii. Specify the partition of training and testing for samples.

Note: An optional approach is random sampling in a certain proportion such as 0.85:0.15 for each class.

iv. Write the 'owndata_train.txt' and 'owndata_test.txt' to assign it.

b. Enter the './pointnet.pytorch.3D/utils' folder and run the 'train_classification.py' script to perform training.

```
>cd ./pointnet.pytorch.3D/utils

>python3 train_classification.py
```

Note: There are several training parameters that could be specified such as the number of epochs, the batch size, and the path of the dataset. Please custom them according to the characteristics of your problem. The well-trained model parameters would be saved in the 'parameter.pkl' file.

c. Copy the 'parameter.pkl' file to the './pointnet.pytorch-pred/utils' folder.

d. Enter the objective folder, and run the 'pred-test.py' file.

```
>cd ./pointnet.pytorch-pred/utils

>python3 pred-test.py > geo_con.txt
```

Note: The desired output would be printed to the 'geo_con.txt' file.

12. Conduct the supervised learning based on the heatmap dataset and the label obtained from unsupervised learning. Troubleshooting 1, 3, 4, and 5.

Note: The heatmap includes the physical deformation information. Using different labels would cause different models (Zhao et al., 2022). Here, we use the G+T+P model for the demonstration of the flow.

a. Transform the heatmap dataset './2Dimages' folder to the format needed by the VGG16 network according to the label file 'label_G+T.txt' (Simonyan and Zisserman, 2015).
   i. Similar to the step 11, open the 'label_G+T.txt' in Excel and sort the sample number by the label, then write the bash script files 'cppic-crumple.bash', 'cppic-fold.bash', 'cppic-interphase.bash', and 'cppic-quasi-flat.bash'.
   ii. Run the above four scripts.

Note: You could obtain the '2Ddataset_pre' folder containing four subfolders in which includes the heatmap pictures corresponding to specific classification.

b. Activate the virtual environment and launch jupyter notebook to run the '2Dtrain.ipynb' file to conduct training.

```
>conda activate tf23

>jupyter notebook
```

Note: Similarly, the training hyperparameters could be custom according to your requirement. The '2D-weight.h5' file containing the well-trained parameters of the model could be obtained later.

c. Run the script 'predict.ipynb' to perform testing.

Note: The predicted classification result would be printed on the screen.

EXPECTED OUTCOMES

Our model provides a framework to resolve the structural complexity of 2D macromolecules and 3D brain models, which utilizes machine learning methods with embedded geometrical, topological, and physical features. The major desired outcomes in our approach outlined in this protocol are the classification of morphology and performance assessment of different models (refer to Zhao et al., 2022 for more details). The following quantification and statistical analysis exhibit the details of evaluation for different models.

QUANTIFICATION AND STATISTICAL ANALYSIS

We choose common metrics for classification tasks including accuracy, precision, recall, and F1 score to evaluate the performance of different models. The accuracy represents the overall predictive

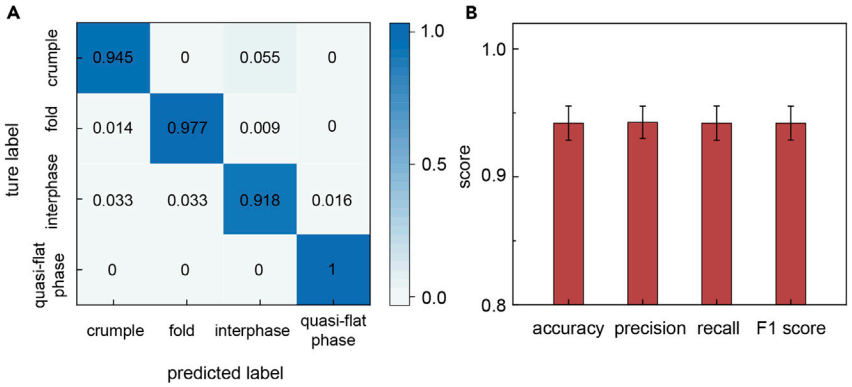


Figure 6. The evaluation of the model

(A) The confusion matrix of morphological phases.

(B) The metrics score of the model capability. The standard deviation calculated from three independent experiments is reported in the error bars.

ability of models which are defined as the ratio of the morphology predicted correctly. The precision characterizes the predictive exactness of models via the percentage of the correct identification in the certainly predicted morphology. The recall measures the capacity of models to recognize the morphology through the ratio of the correct identification in the certainly actual morphology. The F1 score is an integrated metric combining precision and recall which evaluates both the exactness and completeness of models prediction. The combination of these metrics could provide a proper evaluation of the performance of models. From steps 11 and 12, the predicted morphological identification results could be obtained. Combined with the true label, the confusion matrix could be counted and the evaluation metrics could be acquired utilizing call the 'classification_report' function from the sklearn package via the following scripts:

```
>from sklearn.metrics import classification_report

>print(classification_report(y_true,y_pred,digits=4))
```

The predicted results evaluation of G+T+P model are exhibited in Figure 6 and the results of other models could refer to Zhao et al. (2022) for more details.

LIMITATIONS

The dataset format of this protocol is the heatmap and the point cloud, which may limit the scope of the model. When the morphology is not suitable or unable represented by the data format above, the protocol might be unreliable. In addition, the definition of key features might also limit the scope of the protocol. When the conformation could not be distinguished in the defined feature space, the model would be inefficient. Therefore, a theoretical description of morphological complexity including topological features is suggested. However, a general statistic learning framework proposed by this protocol is robust.

TROUBLESHOOTING

Problem 1

Custom the improper parameters in the script, code, or inappropriate dataset format.

Potential solution

Use the default parameters or test the parameters that deviate less from the recommended value. Check the custom data carefully and ensure the format is consistent with our provided dataset.

Problem 2

When running the script in the command prompt, the error 'bash: command not found' occurs.

Potential solution

Make sure to specify the path of the corresponding command and executable files to the environment variable of the system.

Problem 3

When reading the dataset from the specific path or running the script containing path information, the error 'cannot find or open file' is reported.

Potential solution

Check to ensure that files exist on the specified path, and the special characters could not appear in the directory or file name, and check if the file is damaged.

Problem 4

The version incompatibility between different Python packages.

Potential solution

Please refer to the 'before you begin' section, create a virtual environment, and install specific version packages according to the key resources table.

Problem 5

Out of memory during the training process.

Potential solution

Set the batch size parameter to a small value or restart the kernel to free memory.

RESOURCE AVAILABILITY

Lead contact

Further information and requests for resources and reagents should be directed to and will be fulfilled by the lead contact, Zhiping Xu (xuzp@tsinghua.edu.cn).

Materials availability

This study did not generate new unique reagents.

Data and code availability

The data used in this study were generated from molecular simulations and finite element modeling simulations. The codes and data used in the paper are available at https://github.com/zhaoyj21/morphology_complexity. The latest DOI is https://doi.org/10.5281/zenodo.6865044.

ACKNOWLEDGMENTS

This study was supported by the National Natural Science Foundation of China through grants 11825203, 11832010, 11921002, and 52090032. The computation was performed on the Explorer 100 cluster system of Tsinghua National Laboratory for Information Science and Technology.

AUTHOR CONTRIBUTIONS

Conceptualization, Z.X.; methodology, Y.Z.; investigation, Y.Z.; writing, Y.Z. and Z.X.; funding acquisition, Z.X.; supervision, Z.X.

DECLARATION OF INTERESTS

The authors declare no competing interests.

REFERENCES

Hartigan, J.A., and Wong, M.A. (1979). Algorithm AS 136: a K-means clustering algorithm. Appl. Stat. *28*, 100–108. https://doi.org/10.2307/2346830.

Humphrey, W., Dalke, A., and Schulten, K. (1996). VMD: visual molecular dynamics. J. Mol. Graph. *14*, 33–38. 27-28. https://doi.org/10.1016/0263-7855(96)00018-5.

Monga, V., Banerjee, A., and Evans, B.L. (2006). A clustering based approach to perceptual image hashing. IEEE Trans. Inform. Forensic. Secur. *1*, 68–79. https://doi.org/10.1109/TIFS.2005.863502.

Plimpton, S. (1995). Fast parallel algorithms for short-range molecular dynamics. J. Comput. Phys. *117*, 1–19. https://doi.org/10.1006/jcph.1995.1039.

Qi, C.R., Su, H., Mo, K., and Guibas, L.J. (2017). PointNet: deep learning on point sets for 3D classification and segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (IEEE), pp. 652–660. https://openaccess.thecvf.com/content_cvpr_2017/html/Qi_PointNet_Deep_Learning_CVPR_2017_paper.html.

Razavi, M.J., Zhang, T., Li, X., Liu, T., and Wang, X. (2015). Role of mechanical factors in cortical folding development. Phys. Rev. E Stat. Nonlin. Soft Matter Phys. *92*, 032701. https://doi.org/10.1103/PhysRevE.92.032701.

Shrake, A., and Rupley, J.A. (1973). Environment and exposure to solvent of protein atoms. Lysozyme and insulin. J. Mol. Biol. *79*, 351–371. https://doi.org/10.1016/0022-2836(73)90011-9.

Simonyan, K., and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015 (IEEE), pp. 1–14. Conference Track Proceedings. https://dblp.org/rec/journals/corr/SimonyanZ14a.bib.

Wold, S., Esbensen, K., and Geladi, P. (1987). Principal component analysis. Chemometr. Intell. Lab. Syst. *2*, 37–52. https://doi.org/10.1016/0169-7439(87)80084-9.

Zhao, Y., Qin, J., Wang, S., and Xu, Z. (2022). Unraveling the morphological complexity of two-dimensional macromolecules. Patterns *3*, 100497. https://doi.org/10.1016/j.patter.2022.100497.