

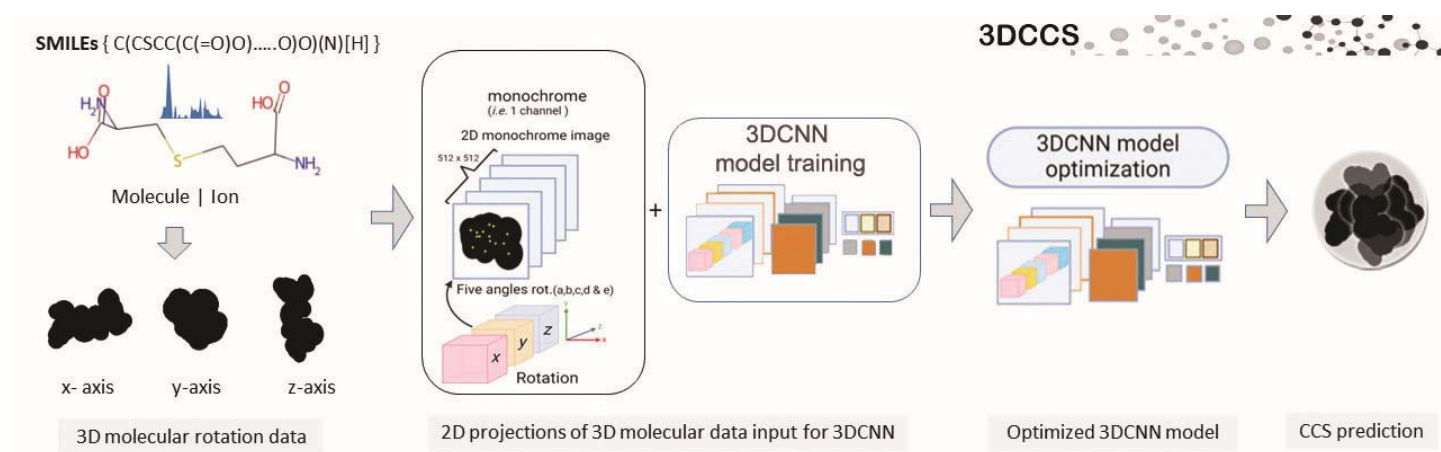
Deep3DCCS

Deep Learning Approach for CCS Prediction from Multi-Angle Projections of Molecular Geometry

Overview

Deep3DCCS is a comprehensive deep learning framework for predicting **Collision Cross-Section (CCS)** values using three-dimensional molecular structure information derived from mass spectrometry data. The framework employs a specialized **3D Convolutional Neural Network (3DCNN)** operating on **multi-angle 2D projections** of molecular geometries. This design bridges molecular structure representation with ion mobility spectrometry, enabling accurate CCS prediction for applications in **metabolomics, lipidomics, and structural biology**.

Schematics of Deep3DCCS



Key Features

Molecular Structure Processing

- Converts **SMILES** into optimized **3D molecular structures**
- Utilizes **Gaussian View 6.0 / Gaussian 16, RDKit**, and computational chemistry methods
- Generates physically realistic geometries suitable for CCS modeling

Multi-View 2D Projections

- Generates rotational 2D projections around **x-, y-, and z-axes**
- Configurable rotational increments (1, 3, 5, or 10 angles per axis)
- Image resolutions from **8×8 to 192×192 pixels**
- Center-of-mass recentering and van der Waals surface visualization

3DCNN Regression Model

- Custom **3D CNN architecture** optimized for CCS regression
- Processes volumetric stacks of multi-angle projections
- Includes 3D convolution blocks, global average pooling, and dense regression layers

Automated Workflow (GUI)

- **PyQt5-based graphical interface**
- End-to-end pipeline from SMILES input to CCS prediction
- Modular workflow tabs:
 - Molecular Structure Optimization and 3D Structure construction
 - Multi-view 2D Projection dataset Generation
 - Model Training /Parameter optimization/ Corss-validation
 - Inference

Comprehensive Evaluation

- Primary metric: **Relative Percentage Error (RPE)**
- Additional metrics: MAE, MAPE, Pearson r, R^2 , standard error
- Automated visualizations:
 - Scatter plots
 - Residual density plots
 - QQ plots
 - Comparative bar charts

Installation

[Recommended] Portable GPU version release for Windows 64bit systems

A portable, GPU-enabled, and self-contained version of Deep3DCCS is available for download as a 7z compressed file.



<https://drive.google.com/file/d/1MNw610K th XiDDnGqTO2lYfzHZs05V7/view?usp=sharing>

Users need to download the 7zip compressed file, decompress using (7zip or winRAR for win64) and run run_deep3DCCS_gpu.bat. All example datasets/models to train/validate are included

Usage

GUI Interface

Launch the main application:

```
python deep3dcnn_main.py
```

General Instruction for basic operation

The Deep3DCCS GUI enables users to visually select and operate three key modules (separated in three major tabs with in the main gui windows):

- Module 1: SMILES Data pre-processing & 2D-projection database generation
- Module 2: 3DCCS model training
- Module 3: Inference using trained model
- All three steps can be summarized below.

Graphical user friendly interface of Deep3DCCS (beta version)

The Deep3DCCS is a complete GUI-feature enabled software toolkit that can be run locally for both training and inference with full customization. Hovering mouse on most of the buttons in GUI interface will provide users short description of its corresponding functions.

Welcome screen for Deep3DCCS gui interface This is a basic welcome screen along with general information about the software. It also links to its github repository.

Deep3DCCS: Geometry-Driven DL Approach for Lipid CCS Prediction from Multi-Angle Molecular Projections

Dataset Processing | Training | Inference
About Deep3DCCS
3DCCS

SMILES { C(CSCC(C(=O)O)....O)O)(N)(H) }

Molecule | Ion

3D molecular rotation data

monochrome (i.e. 1 channel)
2D monochrome image
512 x 512
Five angles rot. (a,b,c,d & e)
Rotation

+

3DCNN model training

3DCNN model optimization

3DCCS

2D projections of 3D molecular data input for 3DCNN

Optimized 3DCNN model

CCS prediction

Deep3DCCS: Geometry-Driven DL Approach for Lipid CCS Prediction from Multi-Angle Molecular Projections

Accurate prediction of collision cross sections (CCS) is critical for molecular identification in ion mobility–mass spectrometry–based lipidomics. Existing CCS prediction approaches often rely on handcrafted molecular descriptors or simplified representations that incompletely capture three-dimensional molecular geometry. Here, we present a deep learning framework that leverages multi-view 2D projections of optimized 3D molecular adduct structures as volumetric inputs to a three-dimensional convolutional neural network (3DCNN). By learning cross-view spatial dependencies directly from molecular shape, the proposed model enables geometry-aware CCS regression across multiple lipid adduct species. This approach provides a scalable and generalizable alternative to conventional CCS prediction tools.

1. Molecular Representation and Multi-View Encoding Strategy

From SMILES to Structured 2D projection inputs

Lipid molecules from the METLIN database and an independent in-house dataset were first standardized into optimized three-dimensional adduct structures starting from SMILES representations. These chemically optimized 3D geometries were then transformed into a machine-learning-compatible format using a **multi-view 2D projection strategy**. Each molecule was spatially normalized and rendered as a stack of orthographic 2D projections generated from systematic rotations along the cartesian axes. This approach converts explicit molecular geometry into structured image volumes that preserve global shape, spatial extent, and orientation-dependent features critical for collision cross-section (CCS) prediction.

2. 3DCNN Architecture for CCS Regression : Learning Cross-View Spatial Dependencies

A **three-dimensional convolutional neural network (3DCNN)** was designed to directly operate on stacked multi-angle projections. Unlike traditional 2D CNNs, the model exploits correlations across rotational views, enabling the network to learn spatial shape descriptors rather than isolated symbolic/linear features. The architecture consists of sequential 3D convolutional blocks that progressively extract hierarchical spatial features, followed by global pooling to preserve holistic structural information while reducing dimensionality. Fully connected layers transform these learned representations into a continuous CCS prediction, framing CCS estimation as a regression problem driven by learned molecular geometry.

network idle: awaiting task

visit repository

0%

The Deep3DCCS data pre-processing module Module 1: Model pre-processing is a straight forward step in GUI window under the Tab "Data pre-processor". Users are required to upload the SMILES datasets (.csv or .xlsx files- Examples of formatted datasets are provided in ".\datasets/" folder).The experimental CCS values is mandatory only for training SMILES dataset pre-processing.

IMPORTANT NOTE: Molecule names may contain Latin or Latin-like encoded characters that can cause issues during preprocessing of .csv or .xlsx files. Therefore, the SMILES input file must always be saved using "latin-1" encoding. In Python, specify the encoding explicitly when writing the file (e.g., with open("SMILES_datafile.csv", "w", encoding="latin-1") as f: if you are creating a custom training or inference SMILES dataset.

Dataset Processing | Training | Inference | About Deep3DCCS

3DCCS

Data pre-processor | Trainer | Inference | System

SMILES-based optimized molecular structure ☐ Adduct Id-based dataset sorting

SMILES dataset (.csv | .xlsx) ...

SDF/ co-ordinates output ...

LIPMETLIN00003
LIPMETLIN00004
LIPMETLIN00006
LIPMETLIN00007
LIPMETLIN00008
LIPMETLIN00010
LIPMETLIN00020
LIPMETLIN00023
LIPMETLIN00024
LIPMETLIN00027
LIPMETLIN00029

Step1: Process structure

☒ Use adduct IDs as labels ☒ Show predicted structure
☐ Use molecule | ion name ☒ Autoset 2D projection path

☒ 3D preview

2D projection extractor from optimized structure

SDF/Co-ordinates folder ...

2D projection output ...

LIPMETLIN00003.txt
LIPMETLIN00004.txt
LIPMETLIN00006.txt
LIPMETLIN00007.txt
LIPMETLIN00008.txt
LIPMETLIN00010.txt
LIPMETLIN00020.txt

Step2: Process 2D projection

No. of Rotations #

☒ Preview 2D projections ☒ Autoset Trainer path

☒ 3D preview

status: 2D projection dataset construction | ETA(min): 0.00000 | ~(epoch, iter., items)/s : 3.6828 | 100%

Shows the concatenated 2D projection preview for X-axis

- This is followed by choosing appropriate options and pressing: "Step1:Process structure" button for 3D optimized structure datafiles generation from SMILES.
- Subsequently, the next step is to produce 2DCCS compatible 2D-projection training/inference-compatible dataset by pressing "Step2: Process 2D projection" button.
- In each case, users must set/select correct number of rotations accordingly.
- Basic preview of the molecule structure and 2D-projections can be seen in the preview window for each dataset processing step.

The Deep3DCCS data Model training module


```

C:\WINDOWS\system32\cmd.exe
base_3dccc_model : ./mode
base_model_flag : False
base_reg_model : D:\dee
model_config : ./mode
store_weights : False
show_model_summary : False
dataset_id for Train|Test : METLIN
eval_dir for Train|Test : METLIN
train_epoch : 250
batch_size : 8
learning_rate : 0.0001
lr_decay_epoch : 250
lr_decay_frac : 0.05
use_lr_decay : True
random_seed : 32437
sample_limit : None
num_rotation : 5
train_singlemode : True
activation_func : linear
loss_func : MSE
relu_alpha : 0.001
set_precision : 4
img_dim : 32
image_threshold : 220
use_threshold_gui : False
use_multipixel_flag : False
min_cutoff_weight : 0
max_cutoff_weight : 1000
post_train_evaluation : True
use_computed_mass : True
AllCCSID_name_flag : True
molecule_name_flag : False
inf_adduct_type : N/A
autoset_missing_exp_ccs : True
sort_raw_SMILE_input_data : False
gpu_index : 0
gpu_name : /physi
set_gpu_growth : False
timestamp : 202602
exp_counter : 3
run_mode : Running
=====
#Reading database...
#Using computed molecular mass from SMILES
Reading database :: 358it [00:00, 454it/s]
#Total Molecules added : 350
#Total Molecules skipped: 8
#Sample folders with CCS: 350 | Missing: 0
#Scanning data :: 100%
#Data shape before extract_mass & mz_ratio : (350, 128)
#Data shape after extract_mass & mz_ratio : (350, 128)
Train: Test: Validation ratio | 0.6

```

Deep3DCCS: Geometry-Driven DL Approach for Lipid CCS Prediction from Multi-Angle Molecular Projections
3DCCS

Dataset Processing | Training | Inference | About Deep3DCCS

Data pre-processor | **Trainer** | Inference | System

SMILES dataset (CSV only) : ./datasets/METLIN_MmHm_5rot.csv

2Dprojection dataset : ./datasets/METLIN_MmHm_5rot_optimized_structure\2D_projections

☒ Train single dataset mode

Adduct type : [M-H]-

☐ Use batch image dimensions : 8,32,64

Train epoch : 250

Batch size : 8

Image dimension : 32

Experiment count : 3

☐ Use base model : ./models/base_model/METLIN_MmHm_5rot_base_model_dkpt.h5

Trained model path : ./models

Evaluation path : ./evaluations

Learning Rate : 0.000100

☒ Decay % : 5 per epoch : 250

☐ Max. cut-off weight : 1000

☒ Min. cut-off weight : 20

☐ Use GUI-based selection for surface binerization : 220

Model activation type : linear

Model loss function : MSE

ReLU alpha value : 0.0010

Taining data (%) : 60

Testing data (%) : 20

Validation data (%) : 20

Random seed : 32437

Data precision : 4

☒ Compute and log exact mass from SMILES

status: Train 3DCNN model | ETA(min): 0.38170 | ~(epoch, iter., items)/s: 5.5439 | 49%

Shows the real-time model losses, FID curves. Right click to get raw data / zoom and other features.

Validation MAPE

Abs. mean validation % error

Shows the real-time validation Mean Average Percentage Error (MAPE) graph

Train: Single | Batch
Train K-fold: Single | Batch
Exit

- Module 2: This module can be accessed by selecting the "Trainer" tab in the main window.
- User will have to provide the SMILES dataset (The same SMILES dataset file which they used to pre-process 2D-projection dataset in "Data pre-processor" module. Only .CSV file is accepted at this point)
- This is followed by selecting 2D projection dataset folder path associated with the SMILES dataset. The software by default sets path for the 2D projection datasets but users can change according to their experimental setup.
- Users can select storage path for the models and evaluations results.
- Users must setup the correct number of rotations and pixel resolution for each training dataset that matches with the training dataset.
- The users can train single or multiple dataset for different pixel resolutions and rotations based on their requirements.
- The option to train single dataset can be selected by tickmarking "Train single dataset mode" checkbox.
- The option to train multiple resolution can be applied by tickmarking "Use batch image dimensions" checkbox.
- The batch or single mode training itself can be done for random experimentation mode by pressing [Train:Single | Batch] or K-fold [Train K-fold: single | Batch] buttons. (For further details, see under title "Optimization of Training Models")
-

The Deep3DCCS data Inference/ccs prediction using trained module

8	Beta-D-mannosyl ...	OC[C@@...]	lipmetlin00244	461.231	210.06836	209.3667
9	08:0 PS	C(=O)([C@...]	lipmetlin00301	510.2473	224.97498	224.9333

Adduct type

[M-H]-

☒ Select the model resolution

32

☒ Select the image rotation(s)

5

Truncate Barchart



100

☒ Max. cut-off weight



1000

☒ Min. cut-off weight



20

☒ Save inference table to evaluation output

☒ Autoselect experimental CCS values to zero if value(s) or feature not found

Inference

Export results

Exit

status: Running sample CCS prediction | ETA(min): 0.00000 | ~(epoch, iter., items)/s : 55.4861 |

100%

Inference results are shown here with key feature values (IDs,PPR, SMILES...)

- Module 3: This is the final inference/prediction module.
- Users will have to provide the optimized/trained model for particular rotation and adduct user "3DCCS model path".
- User will have to set the 2D-projection dataset path associated with the SMILES input datafile under "2D projection dataset" path. The software by default sets path for the 2D projection datasets but users can change according to their experimental setup.
- The experimental CCS values are not mandatory for unknown compounds but are required for conducting external validation.
- Users can input the sample data provided along with the code or their own inference dataset for model validation.
- All SMILES samples for inference/external validation should be processed in the same way as for training dataset.
- Although any pixel resolution is accepted during inferencing for a model trained with particular rotation, it is recommended to use the same pixel rotation that the model was originally trained on.
- Once the path/parameters setup is ready, press "Inference" button to compute the CCS values.
- Finally, inference results along with SMILESs, molecule IDs, name, predicted ccs, experimental ccs (if available), RPE will be displayed in the result table box and stored in the "./evaluation" folder along with other by default. -If user has provided SMILES data with experimental ccs for external validation, the several comparative charts will be generated in the evaluation folder along with ccs prediction results.

The GUI provides:

- **Molecular Optimization:** SMILES → optimized 3D structures
- **2D Projection Generation:** Multi-angle image stacks
- **Model Training:** Architecture and hyperparameter configuration
- **Inference:** CCS prediction on unseen molecules

Command-Line Tools

- **TO DO: in future version** , currently to make the users familiar with all steps and avoid any confusion in parameters setting, only GUI version is available.

Experimental Workflow

Sample pre-processing (for both Training and inference)

Pre-processing Step #1: Select the CSV or XLS files containing sample IDs, SMILES, Adduct information to create standard 3D image dataset **Pre-processing Step #2:** Select the subsequent 3D image dataset to create final 2D training dataset angle projection dataset.

- This 2D projection dataset along with the SMILES dataset file (CSV only) file is used for model training and inference/external validation.

Optimization of Training Models

** There is no golden rule to identify best model training/optimization parameters in deep learning/machine learning. We suggest conducting two step experimentation to: (i) First identify a potentially suitable parameters using random sampling for model training (ii) Using 4-fold cross-validation with the observed most accurate parameters configuration for model optimization. Finally use the optimized model for inference. Additionally, the optimized model parameters can be used to train on entire train/test dataset and further validated on external validation dataset to assess the real-world generalization of the optimized model.

Experiment 1: Resolution & Rotation Optimization

- Resolution range: **8×8 to 192×192**
- Rotations per axis: **1–10**
- 40 replicates per configuration
- Training: 250 epochs, batch size 8, Adam (lr=1e-4)

Experiment 2: 4-Fold Cross-Validation

- Optimal parameters from Experiment 1
- Stratified 4-fold split by adduct
- 15 replicates with different random seeds

NOTE: Although a fixed random seed is used to ensure reproducibility, the inherent complexity of the 3D CNN architecture can lead to minor variations between training runs. Nevertheless, the overall performance trend remains consistent, and differences in inference results are not significant.

Rotation-specific datasets are generated by changing the “Xrot” label in the filename and selecting the same rotation value (e.g., 5rot) in the software. The RAW CSV/XLS files remain identical; only the software’s processing differs.

A single RAW dataset per adduct can generate all rotation variants. Separate datasets are provided here only for demonstration. Based on the selected 2D projection configuration, the software automatically generates the corresponding 3D dataset and 2D projections.

Valid SMILES Input datafile teample (csv file)

A valid templat csv file suitable for both Train and inference/prediction is provided: `./datasets/template.csv`
Note that the column 'exp_ccs' for experimental ccs values are not required for the prediction Note:mz_ratio data is (optional). However, if the user supplies mz_ratio values, it will be included as a part of information metadata. We highly recommend keeping the mz-ratio data for proper documentation purpose in mass spectrometry

Optional: An identical Microsoft XLS file is included for better annotation and customization of the input data.

Model Architecture

- Input: ($N_{rotations} \times H \times W \times Channels$)
 - $4 \times 3D$ convolution blocks ($32 \rightarrow 64 \rightarrow 128 \rightarrow 256$ filters)
 - Batch normalization + anisotropic pooling
 - GlobalAveragePooling3D
 - Dense layers: $128 \rightarrow 64$ (LeakyReLU, Dropout 0.3)
 - Output: Single linear neuron (CCS regression)
 - Optimizer: **Adam (lr=0.0001)**
-

Repository Structure

```
Deep3DCCS/  
├── deep3dcnn_main.py.py  
├── _core.py  
├── utility_modules.py  
├── helper_tools.py  
├── ui_interface.ui  
├── run_deep3DCCS_GPU.bat  
├── run_deep3DCCS_CPU.bat  
├── datasets/  
├── models/  
├── evaluations/  
├── configs/  
├── assets/  
├── requirements-cpu.txt  
└── requirements-gpu.txt
```

NOTE: For inference or prediction on molecules with unknown experimental CCS values, the option “Autoset experimental CCS values to zero if value(s) or feature not found” must be enabled. This forces all experimental CCS values to zero. This option is applicable to CSV SMILES input datafiles that lack the exp_ccs column entirely or contain missing values (e.g., N/A or NULL)for some molecules. In both cases, the scoring metrics reported below are not meaningful, as no comparison with known experimental CCS values is possible.

Evaluation Metrics

- **Relative Percentage Error (RPE)**
- Mean Absolute Percentage Error (MAPE)
- Pearson Correlation (r)
- Coefficient of Determination (R^2)

Interpretation Guide:

- RPE < 3%: Excellent
- 3–5%: Good
- 5–10%: Moderate
 - 10%: Requires refinement or the model is not optimized to deal with such structure due to limited dataset and/or model architecture.

Benchmarking

Deep3DCCS is benchmarked against:

- **CCSBASE**
- **ALLCCS**

Evaluation includes RPE distribution, paired statistical tests, computational efficiency, and external dataset validation.

Output Files

- **3D Structures:** SDF + Cartesian coordinate files
- **2D Projections:** PNG stacks with JSON metadata
- **Models:** HDF5 (.h5) + checkpoints
- **Results:** CSV, JSON summaries, publication-quality plots

CUDA/cuDNN Core installation (for windows 64 bit OS running with NVIDIA GPU)

CUDA Toolkit 11.8/12.x:

https://developer.nvidia.com/cuda-11-8-0-download-archive?target_os=Windows

To install cuDNN 8.00+ Library for CUDA 11.x/12.x Goto NVIDIA's cuDNN download page:

<https://developer.nvidia.com/rdp/cudnn-archive>

and select download cuDNN for 11.x (or 12.x if installing v12.x) (login may be required)

Download link will be dynamically provided for: CUDA Download cuDNN v8.9.7 (December 5th, 2023), for CUDA 11.x

MSVC C and C++ Build Tools redistributable version for GPU/CUDA/cuDNN support

Depending upon the support packages, some windows system might require Microsoft Visual C++ Redistributable (~18 MB instalelr for x64 system)

<https://learn.microsoft.com/en-us/cpp/windows/latest-supported-vc-redist?view=msvc-170#latest-supported-redistributable-version>

CUDA requirement for the GPU version

For systems running on windows, the tensorflow may require CUDA/cuDNN runtime library files to utilize Nvidia GPU which should be installed seperately. After users have installed default python environment manually (or run portable python environment `"./_env/"` to using supplied `run_deep3DCCS_gpu.bat`) follow the instructions in links below to install official CUDA/cuDNN library. A system windows restart is recommended after installation.

Prerequisites

- **Python ≥ 3.7** (tested with TensorFlow-GPU 2.6.0)
- Supported & recommended OS: **Windows 10/11**, Users can try on **Linux Ubuntu** if they can setup similar python environment with Tensorflow and CUDA support
- `_env` (for windows system) will contain all virtual environment required for running Deep3DCCS. the portable version will be all `_env` embedded. Users are required to install CUDA v11.0-v12.6 and cuDNN 8.5+ for proper running of GPU-version

Hardware Recommendations

- **GPU (recommended):** NVIDIA CUDA-compatible GPU (≥ 8 GB VRAM, 16 GB recommended for higher batch size)
- **System RAM:** ≥ 32 GB (can run smoothly in 16 GB system RAM)

Clone Repository

```
git clone https://github.com/vivekmathema/Deep3DCCS.git
cd Deep3DCCS
Deep3DCCS > run_deep3DCCS_gpu.bat (for windows 64bit GPU/portable version) | (linux users are
advised to install python environment and then use Deep3DCCS > python3 deep3dcnn_main.py )
```

Core Dependencies

```
pip install tensorflow-gpu==2.6.0
pip install keras==2.6.0
pip install scipy==1.4.1
pip install rdkit-pypi
pip install pyqt5 matplotlib numpy pandas scikit-learn opencv-python pillow seaborn tqdm colorama
termcolor
```

The requirments can be installed uisng

```
pip install -r requirements-gpu.txt
```

Citation

If you use Deep3DCCS, please cite the paper :

```
@article{deep3dccs2024,  
  title={Deep3DCCS: Deep Learning Approach for CCS Prediction from Multi-Angle Projections of  
Molecular Geometry},  
  author={Siriraj Metabolomics \& Phenomics Center},  
  journal={Journal of Cheminformatics},  
  year={2026},  
  publisher={Springer}  
}
```

License

This project is licensed under the **MIT License**. See `LICENSE` for details.

Contact

Siriraj Faculty Mahidol University, Bangkok 10700, Thailand	Metabolomics of	& Medicine	Phenomics	Center Siriraj	(SiMPC) Hospital
--	---------------------------	--------------------------	------------------	--------------------------	----------------------------

Funding & Acknowledgments

Supported by:

- National Higher Education Science Research and Innovation Policy Council (NXPO)
- PMU-B (Program Management Unit for Human Resource & Institutional Development)

Computational resources provided by the **SiMPC High-Performance Computing Facility**.
