**User guide**

**Requirements and installation**

DeepGP was performed using python (3.8.3, Anaconda distribution version 5.3.1, <https://www.anaconda.com/>) with the following packages: FastNLP (0.6.0), pytorch (1.8.1), torchinfo (1.7.1), transformers (4.12.5), dgl (1.0.1), bidict (0.22.0), pandas (1.0.5) and numpy (1.18.5).

Install these packages by using “pip install” command:

pip install fastNLP

pip install torch

pip install torchinfo

pip install transformers

pip install dgl

pip install bidict

pip install pandas

pip install numpy

These commands will install the latest stable release of the libraries. If you want to install a specific version of the library, you can specify the version number in the command. Take numpy as an example:

pip install numpy==1.18.5

Once the library is installed, you can verify the installation by importing the library in Python and checking the version number:

import numpy as np

print(np.\_\_version\_\_)

**Example data**

All the example data can be found in the “demo\_data” folder.

We recommend adopting a similar data structure for the concurrent processing of multiple datasets. The experimental spectra and search results should be organized in the following directory structure: main\_folder\_name/organism/folder\_name/.

Spectra files:

MSConvert\_mgf\_demo/demo.mgf

*In order to obtain an MGF file, you will need to employ a program or tool with the ability to convert your data into MGF format. Here, we utilize MsConvert from the ProteoWizard Package (version 3.0.11579) with the peak picking setting to convert experimental spectra (raw files) into the Mascot Generic Format (MGF). Additionally, MGF files produced by pGlyco3 have also been successfully tested. When multiple MGF files are stored under a specified folder, they can be automatically retrieved in one go.*

Sequence searching software results:

pGlycoDB-GP-FDR-Pro\_demo.txt

*The pGlycoDB-GP-FDR-Pro\_demo.txt file contains the pGlyco3 results for glycopeptides. In practical application, the pGlycoDB-GP-FDR-Pro.txt file of the pGlyco3 results can be accessed to obtain the desired file. Furthermore, if you have search results from other tools, they can be utilized by converting their format to match that of pGlyco3.*

**Run the release version of DeepGP using the command line**

The entire process contains three steps:

1. Data processing
2. Model training
3. Prediction

The related code files are appropriately numbered for ease of use, such as “1\_dataset\_format.py”.

These scripts are executed in a command-line interface. Advanced users can also adapt these commands for other command-line interfaces.

* **Glycopeptide MS/MS spectra: pre-processing, training, and prediction**

1. Entry to the folder including DeepGP code files.

Users can navigate to the relevant folder using a command such as cd D:\DeepGP\_code. The path “D:\DeepGP\_code” signifies the directory containing the Python scripts for DeepGP.

1. **Pre-processing**: Convert the library search results (.txt) and experimental glycopeptide spectra (.mgf) into files containing spectral data (.csv).

python 1\_dataset\_format.py --datafold D:/DeepGP\_code/demo\_data/human/demo/ --dfname pGlycoDB-GP-FDR-Pro\_demo.txt --mgfdatafold MSConvert\_mgf\_demo --output\_name demo\_data\_1st.csv --dup Drop\_duplicated

The description of the parameters of the command line:

--datafold: This parameter denotes the directory where both the pGlyco3 identification results (pGlycoDB-GP-FDR-Pro.txt) and experimental spectra (.mgf) are stored.

--dfname: This parameter signifies the file name of the pGlyco3 identification results.

--mgfdatafold: This parameter corresponds to the folder name for all .mgf files. (The .mgf files are located within the folder indicated by datafold+mgfdatafold)

--output\_name: This parameter sets the name for the output file.

--dup: This parameter determines the method for removing duplicate identification results. It has three possible values: “Duplicated”, “Drop\_duplicated”, and “Retained\_all”. “Duplicated” only keeps duplicated identification results. “Drop\_duplicated” retains a single identification result, choosing the one with the smallest “TotalFDR” if duplicates exist. “Retained\_all” keeps all identification results. The default is “Drop\_duplicated”.

--mgfsourceorign: This parameter allows you to select the format for .mgf files. The options are “MsConvert” and “pGlyco3”. The default is “MsConvert”.

1. **Model Training:** Train DeepGP model for intact glycopeptide MS/MS prediction

For ease of use, users have the option to utilize our provided trained model for immediate testing, thereby bypassing the model training phase. The trained model can be accessed directly from [DeepGP GitHub Release Page] (<https://github.com/lmsac/DeepGP/releases/>).

For those with access to other datasets, model training can also be conducted using your own datasets or extensive datasets downloaded from public databases. This provides more flexibility and customization, allowing the model to better adapt to various types of data.

python 2\_train\_byBY.py --task\_name demo --folder\_path D:/DeepGP\_code/demo\_data/ --organism human --pattern \*\_data\_1st.csv --trainpathcsv demo/train\_combine.csv --ms2\_method cos\_sqrt --model\_ablation DeepFLR --DeepFLR\_modelpath D:/DeepGP\_code/model/DeepFLR/best\_\_2deepchargeModelms2\_bert\_mediancos\_2021-09-20-01-17-50-729399

The description of the parameters of the command line:

--task\_name: This parameter sets the name for the task.

--folder\_path: This parameter specifies the name of the main folder.

--organism: This parameter identifies the organism of the dataset. Since the data is organized as main\_folder\_name/organism/folder\_name/, data within the specified main\_folder\_name/organism/ will be chosen as the training datasets. Multiple folders can be selected at once. For example, inputting “mouse,human” will select datasets under both main\_folder\_name/mouse/ and main\_folder\_name/human/.

--pattern: This parameter denotes the suffix for the training datasets. Files bearing this suffix within the folder name will be employed as training datasets.

--testdata: This parameter indicates the test data. Files containing the term “testdata” will be omitted from the training files and reserved for further testing. The default is “alltest”.

--trainpathcsv: This parameter represents the output file name for the training datasets. All files within the folder with a specific suffix are combined, processed, and output with this filename.

--ms2\_method: This parameter determines the metric used for DeepGP. Options are “cos\_sqrt”, “cos”, and “pcc”, representing cosine similarity with a square root transformation, cosine similarity, and Pearson correlation coefficient, respectively.

--model\_ablation: This parameter chooses the model to be used. Options include DeepFLR, BERT and Transformer, with the default being “DeepFLR”.

--DeepFLR\_modelpath: This parameter is used for DeepFLR, a pre-trained model previously published and available at [DeepFLR GitHub Release Page](<https://github.com/lmsac/DeepFLR/releases>). Please download it and specify the model path for DeepFLR. If the BERT or Transformer models are used instead of DeepFLR, this parameter can be ignored.

--lr: This parameter adjusts the learning rate, defaulting to 0.0001.

--device: This parameter sets the device number for CUDA. If no GPU is available, the CPU will be used by default. The default device number is 0.

Advanced users can also adjust the settings available in the code’s utilities (utils.py). The model architecture can be easily modified using keywords. For example, if you type “GNN\_global\_ablation=GIN”, you will change the GNN architecture for glycan global representation into GIN. If you type “GNN\_edge\_ablation=GIN”, it means the GNN architecture of glycan B/Y ions intensity prediction is GIN. Users can also change the dimension and layer number by inputting their self-defined number. For example, “GNN\_edge\_num\_layers=7” means that the layer number of GNN for glycan B/Y ions intensity prediction is equal to 7. You can replace “7” with the number of layers you want.

We highly recommend training DeepGP using larger datasets. The demo dataset provided contains only 40 unique spectra. While the code can be successfully implemented with this dataset, it is not large enough to effectively train a model. Therefore, for optimal performance and accuracy, consider using larger datasets.

1. **Prediction**: Predict MS/MS glycopeptide spectra with trained model

python 3\_replace\_predict\_byBY.py --trainpathcsv D:/DeepGP\_code/demo\_data/human/demo/demo\_data\_1st.csv --datafold D:/DeepGP\_code/demo\_data/human/demo/ --bestmodelpath D:/DeepGP\_code/model/human/2023-06-25-18-00-35-186908/epoch-147\_step-28224\_mediancos-0.938205.pt --savename demo --ms2\_method cos\_sqrt --postprocessing off

The description of the parameters of the command line:

--trainpathcsv: This parameter specifies the input file name for the test dataset.

--datafold: This parameter denotes the directory name for the output files.

--bestmodelpath: This parameter sets the model path file name.

--device: This parameter indicates the device number for CUDA. If no GPU is available, the CPU will be used by default. The default device number is 0.

--savename: This parameter provides the prefix for the output file names.

--ms2\_method: This parameter decides the metric used for DeepGP. Options include “cos\_sqrt”, “cos”, and “pcc”. These represent cosine similarity with a square root transformation, cosine similarity, and Pearson correlation coefficient, respectively.

--postprocessing: This parameter determines whether post-processing is required (“on/off”). If set to “on”, the output will include files containing all predicted fragments and their corresponding intensities, along with all experimental fragments and their intensities.

We have uploaded trained models to [DeepGP GitHub Release Page](<https://github.com/lmsac/DeepGP/releases/>). These models are labeled as “human”, “mouse”, and “human&mouse”. The “human” model was trained using human datasets, “mouse” was trained with mouse datasets, and “human&mouse” was trained using a combination of both human and mouse datasets.

It takes less than 2 seconds to predict 40 spectra on a single RTX 3090 GPU.

* **Glycopeptide iRT: pre-processing, calibration, training, and prediction\**

1. Pre-processing

Pre-processing code is the same for the MS/MS prediction.

For the RT calibration process, which should be carried out on multiple datasets, we provide three mouse datasets as examples. The large demo data corresponding to these examples has been uploaded to [DeepGP GitHub Release Page](https://github.com/lmsac/DeepGP/releases/). You can use these datasets as a reference for your own iRT process.

python 1\_dataset\_format.py --datafold D:/DeepGP\_code/data/mouse/PXD005411/ --dfname pGlycoDB-GP-FDR-Pro\_PXD005411.txt --mgfdatafold MSConvert\_mgf\_PXD005411 --output\_name PXD005411\_MouseBrain\_rt\_1st.csv --dup Drop\_duplicated --mgfsourceorign MsConvert

The parameters are the same as aforementioned.

1. RT calibration: Calibrate the retention time using LOWESS based on one dataset.

python 1\_rt\_calibration.py --pattern \*\_rt\_1st.csv --Cali\_csv D:/DeepGP\_code/data/mouse/PXD005411/PXD005411\_MouseBrain\_rt\_1st.csv --folder\_path D:/DeepGP\_code/data/mouse/ --output\_name All\_adjust\_irt.csv

The description of the parameters of the command line:

--pattern: This parameter specifies the suffix for the input datasets. Files with names ending in this suffix within the specified folder will be used as datasets for calibration.

--Cali\_csv: The file path for the reference file used as a basis for calibrating other datasets.

--folder\_path: The folder that contains all the input files and will be used for the output files.

--output\_name: The name of the output file that will store the calibrated retention time for all the files. This output file is for subsequent retention calibration during model training.

Note: RT calibration is performed using the LOWESS method, with the parameters set as frac:0.2, it:2. After calibration, it’s recommended to review the output calibration plot (“RT\_calibration\_plot.pdf”) and tune the calibration parameters if necessary.

Note: For each input file, a corresponding output file with the calibrated retention time is generated, denoted by adding the suffix (“\_adjust.csv”).

1. Model training

For ease of use, users have the option to utilize our provided trained model for immediate testing, thereby bypassing the model training phase. The trained model can be accessed directly from [DeepGP GitHub Release Page](<https://github.com/lmsac/DeepGP/releases/>).

python 2\_train\_rt.py --irt yes --irt\_csv D:/DeepGP\_code/data/mouse/All\_adjust\_irt.csv --task\_name demo\_rt --folder\_path D:/DeepGP\_code/data/mouse/ --pattern \*\_rt\_1st.csv --testdata PXD005411 --device 0 --trainpathcsv mouse\_train\_irt\_1st\_combine.csv --model\_ablation DeepFLR --DeepFLR\_modelpath D:/DeepGP\_code/model/DeepFLR/best\_\_2deepchargeModelms2\_bert\_mediancos\_2021-09-20-01-17-50-729399

The description of the parameters of the command line:

--irt: This parameter determines whether retention time (rt) calibration is to be performed. Use “yes” to enable rt calibration and “no” to skip rt calibration.

--irt\_csv: If rt calibration is performed, this parameter specifies the file path for the calibrated retention time. It should be the same as the file specified in the output\_name of 1\_rt\_calibration.py. If rt calibration is not performed, this parameter can be omitted.

--folder\_path: The path to the folder that contains the input and output files.

Other parameters are the same as those used in 2\_train\_byBY.py.

(3) Model testing

python 3\_replace\_predict\_rt.py --datafold D:/DeepGP\_code/data/mouse/PXD005411/ --trainpathcsv D:/DeepGP\_code/data/mouse/PXD005411/PXD005411\_MouseBrain\_rt\_1st.csv --device 1 --bestmodelpath D:/DeepGP\_code/model/mouse\_rt/2023-07-14-11-10-12-102196/epoch-48\_step-13536\_r2-0.954393.pt --savename test\_irt\_adjust\_PXD005411

The description of the parameters for the command line is consistent with those used in 3\_replace\_predict\_byBY.py.

For retention time prediction, the “mouse\_rt” model has been specifically trained using mouse datasets. This model is utilized in the current process for making retention time predictions. We have uploaded this trained model to [DeepGP GitHub Release Page](<https://github.com/lmsac/DeepGP/releases/>).

It takes 38 seconds to predict iRT for 6254 spectra on a single RTX 3090 GPU.

**All the demo data and code are provided. A sentence with gray background indicates it is a sentence of code. If you have any further questions, please don't hesitate to ask us via:** [**liang\_qiao@fudan.edu.cn**](mailto:liang_qiao@fudan.edu.cn)**. You could also go to the homepage of DeepGP on GitHub to ask a question.**