DNNGP User Manual

DNNGP, a deep neural network-based method for genomic prediction using multi-omics data in plants

Authors: Kelin Wang, Muhammad Ali Abid, Awais Rasheed, Jose Crossa, Sarah Hearne, **Huihui Li***

Version 3.0

Encoding: UTF-8

2023-05-12

Citation: Wang K, Abid MA, Rasheed A, Crossa J, Hearne S, Li H. DNNGP, a deep neural network-based method for genomic prediction using multi-omics data in plants. Mol Plant. 2023 Jan 2;16:279-293.

Doi: 10.1016/j.molp.2022.11.004, PMID: 36366781

Contact us

Huihui Li: lihuihui@caas.cn

Contents

1. Overview of the DNNGP project ..... - 1 -

1.1 Project address: https://github.com/AIBreeding/DNNGP ..... - 1 -

1.2 File directory structure ..... - 1 -

2. Data preparation ..... - 3 -

3. DNNGP environment setup ..... - 3 -

4. Enter the data file ..... - 4 -

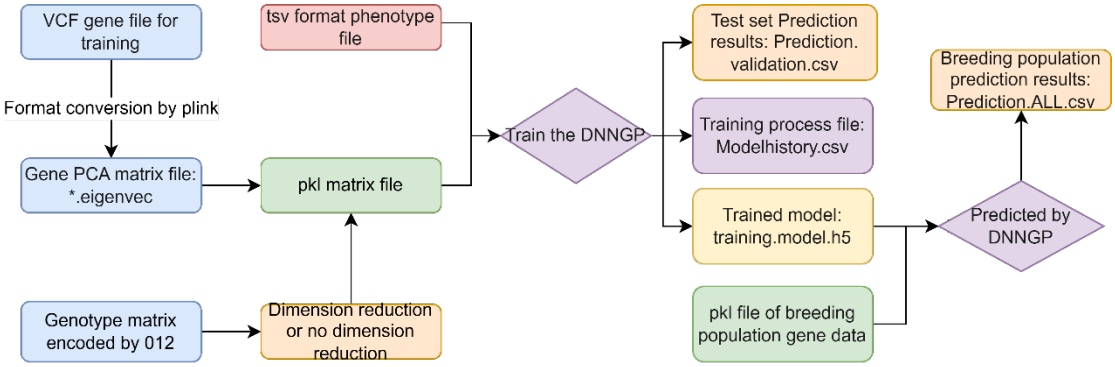
5. DNNGP model training ..... - 5 -

6. Use the trained model to make predictions on the test data ..... - 7 -

7. Special Instructions: ..... - 8 -

1. Overview of the DNNGP project

DNNGP is a genome-wide prediction model based on deep learning theory, which aims to predict plant and animal phenotypes using genome-wide markers. In addition, DNNGP can be used for multiomics data prediction in plants and animals. This model mainly uses Python 3.9.16 and TensorFlow 2.6.0 writes. The training and prediction process of DNNGP is as follows:



1.1 Project address: https://github.com/AIBreeding/DNNGP

Due to Apple's official security policy, the software may undergo several security verifications during operation.

1.2 File directory structure

```
DNNGP:
│   bash_me_first.sh
│   bash_Start_DNNGP.sh
│   CN 使用说明.pdf
│   EN usermanual.pdf
│   requirements.txt
│
├─Input_files
│       tsv2pkl.py
│       tsv2pklGUI
│       wheat1.tsv
│       wheat599_pc95.pkl
│       wheat599_pc95.tsv
│
├─Output_files
└─Scripts
        config_dnngp.cpython-39-x86_64-linux-gnu.so
        DNNGP
        DNNGP-PreGUI.py
        dnngp.cpython-39-x86_64-linux-gnu.so
```

dnngp_runner.py
environment.yaml
Pre-Batch_run.py
Pre_config_dnngp.cpython-39-x86_64-linux-gnu.so
Pre_dnngp.cpython-39-x86_64-linux-gnu.so
Pre_runner.py
The Train_Batch_run.py

The file mainly contains the following five parts:

(1) GUI file

This part of the file mainly includes three files, namely: bash_me_first. sh, Input_files/ tsv2pklGUI, bash_me_run_DNNGP.sh. Under the Linux platform, bash starts bash_click_me_first.sh can be used to build the environment. Double-click Input_files/tsv2pklGUI can convert tsv files to pkl files according to GUI prompts. Then bash started bash_me_run_DNNGP.sh can train and predict models.

(2) requirements.txt / environment.yaml

For environment construction, environment configuration required packages and their versions.

(3) Input_files

In this directory are sample files for input data.

(4) Scripts

This directory contains the scripts needed to train the model and predict the scripts.

Train the model: After completion, the terminal displays the model prediction results to evaluate the training effect. Three files are output at the same time: the trained model (training.model.h5) and the validation set prediction (Prediction.validation.csv), as well as the values of each epoch in the training process (Modelhistory.csv）.

Model prediction: By reading the model trained in the previous step, the phenotype of the breeding population is predicted and the predicted value of all individuals is output (Prediction.ALL.csv).

(5) Output_files

This directory contains the output files after the DNNGP method executes the input sample files.

The use of models should follow the following order: (1) set up the running environment (2) prepare data, (3) train the model, and (4) use the trained model to make predictions

2. Data preparation

Genotypic data processing based on Plink2 software

```
./plink2 --threads 30 --vcf *.vcf --pca 10 --out pca10
```

--threads 30 uses 30 threads

--vcf *.vcf read vcf files

--pca 10 takes PC1-PC10 (can be set ≤ number of samples≤ 8000).

--out pca10 output file name is pca10

If a non-numeric chromosome number is present, the --allow-extra-chr parameter needs to be added

```
./plink2 --allow-extra-chr --threads 30 --vcf *.vcf --pca 10 --out pca10
```

The result is two files with the suffixes .eigenval and .eigenvec, which show the weight of each PC, the weight/gravity of each PC, and the degree of interpretation for each PC. eigenval is the PCA matrix we need to use.

Tips:

(1) The above commands apply to Powershell terminals under Windows platform and Linux and Mac terminals. If you use the cmd terminal under the Windows platform, please replace ./plink2 with plink2.

(2) If PCA is used to convert genetic data, it is necessary to first combine the data of the breeding population and the training population and then perform PCA analysis, and then separate the two after obtaining the PCA matrix.

3. DNNGP environment setup

(1) Download project address: https://github.com/AIBreeding/DNNGP

(2) To run DNNGP, you first need to set up the operating environment:

Install Miniconda (https://docs.conda.io/en/latest/miniconda.html) first and add it to the system environment. (Miniconda can be checked when installing it, which is required to use the GUI.)

Under the Mac platform, you can directly build the environment with one click through the bash bash_click_me_first.sh command.

If your chip is an Intel series chip, use bash bash_click_me_first.sh I

If your chip is an M-series chip, use bash bash_click_me_first.sh M

If the one-click environment failure to build, use the following command to build the running environment:

- conda create -n DNNGP3 python=3.9.16
- conda activate DNNGP3
- cd dnngp
- conda install --yes --file requirements.txt

4. Enter the data file

After the environment is set up, you need to prepare various data files according to the sample data format, which is located in the following directory:

.. /DNNGP/input_file/

It contains the following four files:

- **wheat1.tsv:** A tab-delimited phenotypic data file.
- **wheat599_pc95.tsv:** A tab-delimited principal component matrix file.
- **wheat599_pc95.pkl:** A principal component matrix file that the model can read.
- **tsv2pkl.py:** Convert the format from tsv to PKL file conversion script**.**

Among them, wheat599_pc95.pkl files can be run by **wheat599_pc95.tsv** files by **running tsv2pkl.py (tsv2pkl)** converted. Eigenval files generated by plink2 can also be directly converted via **tsv2pkl.py**.

The conversion method is as follows

After opening the tsv2pkl.py, modify the file path in the fifth line of inpath and the sixth line of outpath to your own file path. Then run tsv2pkl.py DNNGP3 environment created by conda.

python tsv2pkl.py

Under the Mac platform, you can double-click the tsv2pkl program in the Input_files directory to use the GUI interface to complete the file format conversion.

The phenotypic data text file formats are as follows:

ID	env1
M1	1.67162948
M2	-0.25270276
M3	0.341815127
M4	0.785439489
M5	0.998317613
M6	2.336096876
M7	0.617410817

The principal component matrix TSV file format is as follows:

ID	PC1	PC2	PC3	…
M1	7.0408269	2.053877771	-6.161150675	…
M2	5.924749016	1.137903031	1.132296531	…
M3	5.953045926	1.082444715	1.139961515	…

5. DNNGP model training

This section requires two inputs, the principal component matrix file prepared in the previous step and the phenotypic data file. For the specific format, please refer to the previous section.

Parameter description:

--batch_size the sample size called to train the model

--lr initial learning rate

--epoch number of iterations

--dropout1 first feature discard (to prevent overfitting).

--dropout2 second feature discard (to prevent overfitting).

--patience no improvement reduces the learning rate threshold (when the model reaches the threshold without improvement, it automatically decreases the learning rate).

--seed random seed

--cv sets the number of cross-validation folds

--part Select a piece of cross-validation data as the validation set

--earlystopping if the training effect does not improve, the training threshold will be stopped (when the model reaches the threshold without improving the number of times, the training will be automatically stopped and the best parameter will be saved)

--snp principal component matrix file path

--pheno phenotypic data file path

--output output directory

The above parameters, except --snp、-- pheno, and --output, are numeric.

Go to the Scripts directory example command: (Enter different paths depending on your platform type)

```
cd Scripts #Windows, Linux, Mac-Intel series chip platform program path
cd ./Scripts/M1 #Mac-M series chip platform program path
```

Train model example command:

```
python dnngp_runner.py --batch_size 28 --lr 0.001 --epoch 100 --dropout1 0.5 --dropout2 0.3 --patience 5 --seed 123 --cv10 --part 1 --earlystopping 10 --snp "../input_files/wheat599_pc95.pkl" --pheno "../input_files/wheat1.tsv" --output "/Your_path/"
```

Under the Mac platform, you can directly start the GUI through the bash bash_Start_DNNGP.sh command and operate according to the GUI prompts. The right side of the GUI outputs standard commands and the training process.

If your chip is an Intel series chip, use bash bash_Start_DNNGP.sh I

If your chip is an M-series chip, use bash bash_Start_DNNGP.sh M

Train the model output file

After the training is completed, three output files will be generated in the specified output directory, namely:

Prediction.validation.csv: The prediction result of the DNNGP model on the validation set (the ordinal number of the first column represents the name of the predicted value individual in the original data set).

training.model.h5: The trained model file is used for the next step of phenotypic trait prediction in the breeding population.

Modelhistory.csv: records the changes in various values during the training process.

After the training is complete, the terminal displays the Pearson correlation coefficient between the predicted value and the true value as follows:

'Corrobs vs pred =',(0.582, 0.001)

The first number is the correlation coefficient (0.582) and the second number is the *p*-value (0.001).

6. Use the trained model to make predictions on the test data

After getting the trained model file, we want to predict the phenotypic traits of the test data set (i.e., the breeding population). This part requires two input files, one is the model file generated by the previous training step, that is, training.model.h5, and the second is the breeding population principal component matrix file (*.pkl). The format is the same as the one used to train the model in the previous step.

Predict the breeding population phenotypic traits example command:

```
python Pre_runner.py --Model "/Your_path/training.model.h5" --SNP "/Your_path/wheat599_pc95.pkl" --output "/Your_path/"
```

Description of DNNGP prediction parameters:

--Model: the path to the .h5 model file generated when training the model

--SNP: the genetic data file path of the dataset to be predicted

--output: the directory where the prediction result file is generated

Under the Mac platform, you can start the GUI interface (bash bash_Start_DNNGP.sh) through the previous step, and then perform predictive operations according to the GUI prompts.

Model prediction output file

After the DNNGP model completes the prediction, the result file **Prediction.ALL.csv** is generated in the specified directory, which is the phenotypic trait prediction results of all individuals in the breeding population.

7. Special Instructions:

The Script directory contains the name Train-Batch_run.py and Pre-Batch_run.py's Python script can be tested in batches, where the former is a batch script for training a model, and the latter is a batch script for model prediction.

Run the sample command:

python Train-Batch_run.py or python Pre-Batch_run.py