Instructions to run dPL+HBV models

This release contains the codes and related data to train models with differentiable parameter learning (dPL) applied to HBV backbone as shown in papers below. If the code is useful for your work, please cite the following papers.

Feng, D., Liu, J., Lawson, K., & Shen, C. (2022). Differentiable, learnable, regionalized process-based models with multiphysical outputs can approach state-of-the-art hydrologic prediction accuracy. *Water Resources Research*, 58, e2022WR032404. https://doi.org/10.1029/2022WR032404

Feng, D., Beck, H., Lawson, K., & Shen, C. (2022). The suitability of differentiable, learnable hydrologic models for ungauged regions and climate change impact assessment. *Hydrology and Earth System Sciences Discussions*, 1-28, accepted. https://doi.org/10.5194/hess-2022-245

This release will also be merged into our MHPI hydroDL tool, please follow this github link (https://github.com/mhpi/dPLHBVrelease) to get updates. A new clearer version of dPL codes with much less files will also come soon and please check this release in the future.

Bug Fix Doc: This online doc (*https://bit.ly/3TOKmqK*) logs bug fixing information in historical versions of this release and solutions to some common questions. It's good to always check if the listed bugs have been fixed in your downloaded codes before running our examples, especially when your local codes are from older versions. Please also use this file as a reference to find potential solutions if you have trouble running our codes.

If you have any question for this release, please contact us by *duf328@psu.edu* or *cshen@engr.psu.edu*

Follow the below instructions to run the dPL models:

1. Set up the environment

The environment we are using for the above papers is provided as the file `environment.yml` under the release directory *dPLHBVrelease*. To create the same conda environment, please have conda installed and then run:

```
conda env create -f environment.yml
```

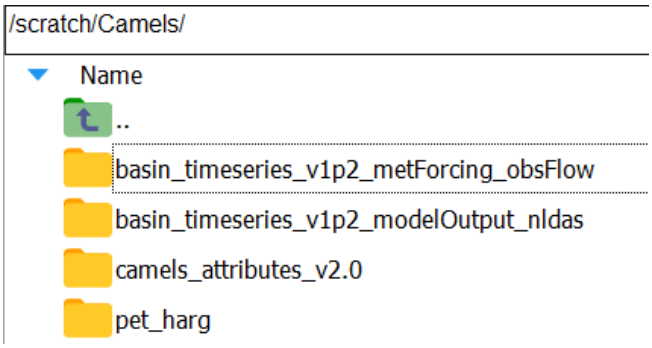
Activate and make sure to use this installed environment to run the codes:

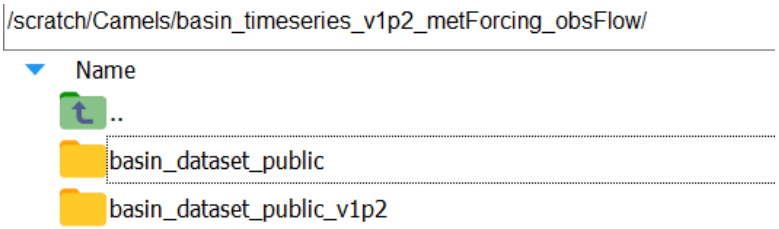
```
conda activate mhpihydrodl
```

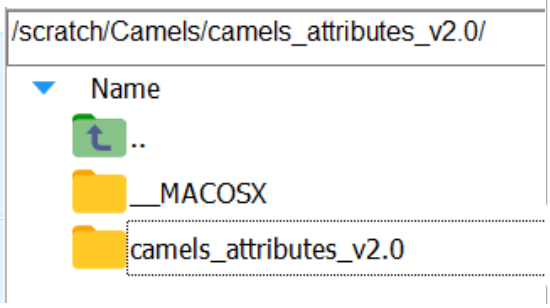
Note: If you have issue for installing the environment, check if you have set the conda channel priority as 'strict'. Change the channel priority as 'flexible' and try again.

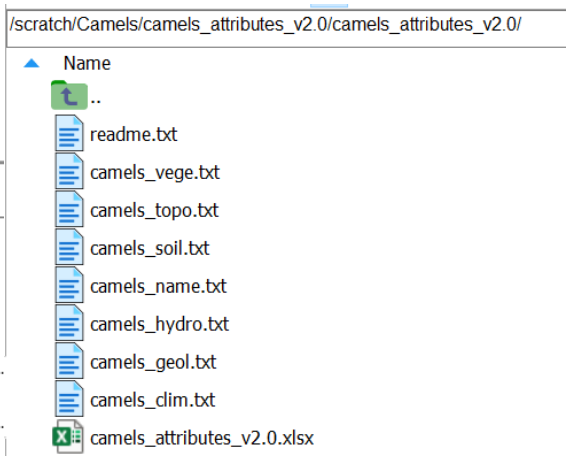
2. Prepare the related data

The dataset used is NCAR **CAMELS** dataset. Download CAMELS following this link (*https://ral.ucar.edu/solutions/products/camels*). We noticed that the original CAMELS structure on this website has been changed recently. Please make sure downloading forcing and discharge observation data *basin_timeseries_v1p2_metForcing_obsFlow.zip* and basin attributes - all the *camels_xxx.txt* files (in previous version organized as one *CAMELS Attributes (.zip)* file). Unzip your files and organize them as these path format */your/path/to/Camels/basin_timeseries_v1p2_metForcing_obsFlow/*, and */your/path/to/Camels/camels_attributes_v2.0/camels_attributes_v2.0/*. Please see the below figures for examples.









You need to set your own directory path *'/your/path/to/Camels'* where you save CAMELS data in your device (for example *'/scratch/Camels'* in our case in the above figures) as the variable *"rootDatabase"* inside the codes later.

You also need the calculated **PET** data to run the models. We have wrapped the PET data along with this code release named *"pet_harg.zip"*. Please unzip and save the "*pet_harg*" directory to the location "*rootDatabase" (/your/path/to/Camels)* as shown in the above first figure.

3. Run the codes

Reminder: if you have errors when running the below codes, check this bug fix log (https://bit.ly/3TOKmqK) which may provide potential solutions.

There are three interface codes as the tutorials for dPL+HBV models as located in "***dPLHBVrelease/hydroDL-dev/example/dPLHBV/***":

1. ***traindPLHBV.py***: the interface to train dPL+HBV models with static or dynamic parameters
2. ***testdPLHBV-Static.py***: the interface to test the trained dPL+HBV models with static parameters
3. ***testdPLHBV-Dynamic.py***: the interface to test the trained dPL+HBV models with dynamic parameters

To repeat core model results in *Feng et al., 2022 WRR* and *Feng et al., 2022 HESSD*, here are some detailed setups to follow.

Path settings: as described in data preparation, set *rootDatabase* as *'/your/path/to/Camels'* in line 85 in *traindPLHBV.py* based on your situation, and *rootOut* (line 88) as the root directory where you define to save your trained models. Also check line 205 *PETDir* for the correct PET data path. Please make sure also applying these path variable settings to two other testing codes *testdPLHBV-Static.py* and *testdPLHBV-Dynamic.py*.

***Feng et al., 2022 WRR* train the models on all basins doing temporal generalization tests. To reproduce the results in that paper:**

1. In *traindPLHBV.py*, set *PUOpt = 0* (line 31) to train on all basins; set *TDOpt = False* or *TDOpt = True* (line 41) to train a model with static parameters or dynamic parameters (dynamic β, γ), respectively; Set *forType = 'daymet'* (line 43); Set *buffOpt = 0* (line 36); Set the training period *Ttrain = [19801001, 19951001]* (line 52) and *Tinv = [19801001, 19951001]* (line 54) as described in the WRR paper. Run this file to train the model.

2. After training finished, use *testdPLHBV-Static.py* to test models with static parameters:

Check the settings from line 32 to 52 to keep same as what you have defined when training the model in order to find the correct path *testsave_path* (line 114) and *testout* (line 142) for testing. Set testing period *Ttest = [19951001, 20101001]* (line 55) and *TtestLoad = [19951001, 20101001]* (line 58); Make sure setting *TestBuff = xTrain.shape[1]* (line 263) and *PUOpt = 0* (line 33) for training and testing on all basins. Run this file to get testing results.

3. Use *testdPLHBV-Dynamic.py* to test models with dynamic parameters:
   Check the setting from line 32 to 57 to keep same as what you have defined when training the model in order to find the correct path *testsave_path* (line 119) and *testout* (line 146) for testing. Set testing period *Ttest = [19951001, 20101001]* (line 55) and *TtestLoad = [19951001, 20101001]* (line 58); Make sure setting *TestBuff = xTrain.shape[1]* (line 269) and *PUOpt = 0* (line 33) for training and testing on all basins. Run this file to get testing results.

Feng et al., 2022 HESSD train the models for Prediction in Ungauged Basins (PUB—randomly hold out basins for test) and Prediction in ungauged regions (PUR – hold out entire regions for test) as spatial generalization tests. To reproduce the results there:

1. In *traindPLHBV.py*, set *PUOpt = 1* or *PUOpt = 2* (line 31) to train models for PUB or PUR tests, respectively; In Feng HESSD we were using “maurer” forcing so setting *forType = 'maurer'* (line 43) if you want to repeat the results; Set *TDOpt = False* or *TDOpt = True* (line 41) to train a model with static parameters or dynamic parameters (dynamic β, γ), respectively; *testfoldInd = 1* (line 91) defines which fold to hold out during training, and you need to modify this each time as one number from 1 to 10 for PUB (10 folds) and 1 to 7 for PUR (7 folds) to finish cross validation training on all basins. Set *buffOpt = 2* (line 36); Set the training period *Ttrain = [19891001, 19991001]* (line 53) and *Tinv = [19891001, 19991001]* (line 55) as described in the HESSD paper. Run this file to train the model.

2. After cross validation training finished, use *testdPLHBV-Static.py* to test models with static parameters for all folds:
   Check the setting from line 32 to 52 to keep same as what you have defined when training the model in order to find the correct path *testsave_path* (line 114) and *testout* (line 142) for testing. Set testing *period Ttest = [19891001, 19991001]* (line 56) and *TtestLoad = [19801001, 19991001]* (line 59, here we use 1980-1989 forcings only to warm up the state variables for PUB/PUR testing); Make sure setting *TestBuff = len(TtestLoadLst) - len(TtestLst)* (line 264) and *PUOpt = 1* or *PUOpt = 2* (line 33) for PUB or PUR testings, respectively. Run this file to get testing results.

3. Use *testdPLHBV-Dynamic.py* to test models with dynamic parameters for all folds:

Check the setting from line 32 to 57 to keep same as what you have defined when training the model in order to find the correct path *testsave_path* (line 119) and *testout* (line 146) for testing. Set testing period *Ttest = [19891001, 19991001]* (line 56) and *TtestLoad = [19801001, 19991001]* (line 59, here we use 1980-1989 forcings only to warm up the states for PUB/PUR testing); Make sure setting *TestBuff = len(TtestLoadLst) - len(TtestLst)* (line 264) and *PUOpt = 1* or *PUOpt = 2* (line 33) for PUB or PUR testings, respectively. Run this file to get testing results.