Instructions to run Mass-Conserving Architectures

Code release created by Yuan-Heng Wang (yhwang0730@gmail.com)

This release contains the codes and data to train Mass-Conserving Architectures (MAs) as summarized in the submitted manuscript below.

Wang, Y.H., & Gupta, H.V. (2024). Towards Interpretable Physical-Conceptual Catchment Scale Hydrological Modeling using the Mass-Conserving-Perceptron. *Water Resources Research*, Submitted for Publication.

If you have any questions about this release, please contact us at yhwang0730@gmail.com | hoshin@arizona.edu

Follow the below instructions to run the MAs:

1. Set up the environment

The environment we are using for the above manuscript is provided in the file environment.yml.

To create the same conda environment, please have conda installed and then run:

conda env create -f environment.yml

Activate the installed environment to run the codes:

conda activate pytorchenv

The code will work when the user applies other Python environments with different package versions such as higher PyTorch versions, if all the necessary packages are installed.

2. Introduction of Each Folder

a. 20220527-MDUPLEX-LeafRiver: Contain the 40-year leaf river data and the associated flag for the training/selection/testing set.

b. MCPBRNN_lib_tools: The main library used to train mass-conserving perceptron and several physics-based or data-driven benchmarks in this study.

c. Training_Script: Each .py file in this folder is used to train a specific case presented in this study.

mcpbrnn_Main_Trial1_Iter1.py: MA_1 architecture with a single cell state (node) and a single flow path (link)

mcpbrnn_Main_PETconstraint_MCA1_BYPASSM0.py: MA_1BP_1 case mentioned in the paper that is MA_1 architecture with first type of input-bypass gate.

mcpbrnn_Main_PETconstraint_MCA1_BYPASSM1.py: $\boldsymbol{MA_1}BP_2$ case mentioned in the paper that is MA_1 architecture with second type of input-bypass gate.

mcpbrnn_Main_PETconstraint_MCA2.py: MA_2 architecture with a single cell state (node) and two flow paths (link)

mcpbrnn_Main_PETconstraint_MCA2_BYPASSM0.py: MA_2BP_1 case mentioned in the paper that is MA_2 architecture with first type of input-bypass gate.

mcpbrnn_Main_PETconstraint_MCA2_BYPASSM1.py: MA_2BP_2 case mentioned in the paper that is MA_2 architecture with second type of input-bypass gate.

mcpbrnn_Main_PETconstraint_MCA3.py: MA_3 architecture with two cell states (node) and a single flow path (link)

mcpbrnn_Main_PETconstraint_MCA3_Constant.py: Same as **mcpbrnn_Main_PETconstraint_MCA3.py** with constant output gate in the routing node. Later the statistics of cell state in the routing tank are used to standardize the variable output gate for the in **mcpbrnn_Main_PETconstraint_MCA3.py.**

mcpbrnn_Main_PETconstraint_MCA3_Variant_BYPASSM0.py: MA_3BP_1 case mentioned in the paper that is MA_3 architecture with first type of input-bypass gate.

mcpbrnn_Main_PETconstraint_MCA3_Variant_BYPASSM1.py: MA_3BP_2 case mentioned in the paper that is MA_3 architecture with second type of input-bypass gate.

mcpbrnn_Main_PETconstraint_MCA4.py: MA_4 architecture with two cell states (node) and two flow paths (link).

mcpbrnn_Main_PETconstraint_MCA4_BYPASSM0.py: MA_4BP_1 case mentioned in the paper that is MA_4 architecture with first type of input-bypass gate.

mcpbrnn_Main_PETconstraint_MCA4_BYPASSM1.py: MA_4BP_2 case mentioned in the paper that is MA_4 architecture with second type of input-bypass gate.

mcpbrnn_Main_PETconstraint_MCA4_GWMR_CellDependent.py: $MA_4MR_{GW}^{\sigma}$ case mentioned in the paper and the $\tilde{c}_{MR_{gw}}$ value is allowed to be either positive or negative.

mcpbrnn_Main_PETconstraint_MCA4_GWMR_CellDependent_NoRelax.py: $MA_4MR_{GW}^{\sigma}$ case mentioned in the paper with $\tilde{c}_{MR_{gw}} \geq 0$

mcpbrnn_Main_PETconstraint_MCA4_constant.py: Same as **mcpbrnn_Main_PETconstraint_MCA4.py** with constant output gate in the routing node. Later the statistics of cell state in the groundwater tank are used to standardize the variable output gate for the in **mcpbrnn_Main_PETconstraint_MCA4.py.**

mcpbrnn_Main_PETconstraint_MCA5.py: MA_5 architecture with three cell states (node) and two flow paths (link).

mcpbrnn_Main_PETconstraint_MCA5_BYPASSM0.py: MA_5BP_1 case mentioned in the paper that is MA_5 architecture with first type of input-bypass gate.

mcpbrnn_Main_PETconstraint_MCA5_BYPASSM1.py: MA_5BP_2 case mentioned in the paper that is MA_5 architecture with second type of input-bypass gate.

mcpbrnn_Main_PETconstraint_MCA5_GWMR_CellDependent.py: $MA_5MR_{GW}^{\sigma}$ case mentioned in the paper and the $\tilde{c}_{MR_{gw}}$ value is allowed to be either positive or negative.

mcpbrnn_Main_PETconstraint_MCA5_GWMR_CellDependent_NoRelax.py: $MA_5MR_{GW}^{\sigma}$ case mentioned in the paper with $\tilde{c}_{MR_{gw}} \geq 0$

mcpbrnn_Main_PETconstraint_MCA6.py: MA_6 architecture with three cell states (node) and three flow paths (link).

mcpbrnn_Main_PETconstraint_MCA6_BYPASSM0.py: MA_6BP_1 case mentioned in the paper that is MA_6 architecture with first type of input-bypass gate.

mcpbrnn_Main_PETconstraint_MCA6_BYPASSM1.py: MA_6BP_2 case mentioned in the paper that is MA_5 architecture with second type of input-bypass gate.

mcpbrnn_Main_PETconstraint_MCA6_GWMR_CellDependent.py: $MA_5MR_{GW}^{\sigma}$ case mentioned in the paper and the $\tilde{c}_{MR_{gw}}$ value is allowed to be either positive or negative.

mcpbrnn_Main_PETconstraint_MCA6_GWMR_CellDependent_NoRelax.py: $MA_6MR_{GW}^{\sigma}$ case mentioned in the paper with $\tilde{c}_{MR_{gw}} \geq 0$

d. Evaluation_Script: Each .py file in this folder is used to evaluate a specific case presented in this study. The file names are the same with the files in the **Training_Script** folder with additional "_EVAL" attached at the end of the original file name.

In short, the reason for providing .py file for each specific case is because the parameter inheritance for each model is unique.

3. Run the codes
The user will need to move one of the training scripts from Training_Script folder back into the parent directory and then running the script.

The user is expected to go into each script and set up:
a. job folder name through updating the "CaseName" variable in the script.
b. running directory through updating the "parent_dir" variable in the script.
c. the directory for parameter initialization through updating the "Ini_dir*" relevant variables in the script.

The user can output all the time series generated by each case by moving the associated files from Evaluation_Script folder back into the parent directory and then running the script.

The user is expected to go into the script and set up:
a. running directory through updating the "parent_dir" variable in the script.
b. the directory where the pt. file result was stored by updating the "Ini_dir*" relevant variables in the script.

The user is expected to go into the script and set up the running directory by updating the "parent_dir" variable in the script.