Instructions to run Mass-Conserving Perceptron

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

This release contains the codes and data to train Mass-Conserving Perceptron (MCP) as shown in the submitted manuscript below.

Wang, Y.H., & Gupta, H.V. (2023). A Mass-Conserving-Perceptron for Machine-Learning-Based Modeling of Geoscientific Systems. *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 MCP:

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, as long as 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_constantO_constantL.py: Single MCP node with constant output and loss gate ($MC\{O_\kappa L_\kappa\}$ in session 4.1).

mcpbrnn_Main_constantO_variableL.py: Single MCP node with constant output and time-variable loss gate ($MC\{O_\kappa L_\sigma\}$ in session 4.1).

mcpbrnn_Main_variableO_constantL.py: Single MCP node with constant loss and time-variable output gate ($MC\{O_\sigma L_\kappa\}$* in session 4.1).

mcpbrnn_Main_variableO_constantL_Trial1.py: Same as **mcpbrnn_Main_variableO_constantL.py** with mean/standard deviation of cell state updated ($MC\{O_\sigma L_\kappa\}$ in session 4.1).

mcpbrnn_Main_variableO_constantL_Trial1_Iter1.py: Same as **mcpbrnn_Main_variableO_constantL.py** with mean/standard deviation of cell state updated from the results obtained through **mcpbrnn_Main_variableO_constantL_Trial1.py** ($MC\{O_\sigma L_\kappa\}$ in session 4.1).

mcpbrnn_Main.py: Single MCP node with time-variable output and loss gate without scaling cell state values ($MC\{O_\sigma L_\sigma\}$* in session 4.1).

mcpbrnn_Main_Trial1.py: Single MCP node with time-variable output and loss gate with mean/standard deviation of cell state updated through the results obtained in **mcpbrnn_Main.py** ($MC\{O_\sigma L_\sigma\}$ in session 4.1).

mcpbrnn_Main_Trial1_Iter1.py: Same as **mcpbrnn_Main_Trial1.py** with mean/standard deviation of cell state updated ($MC\{O_\sigma L_\sigma\}$ in session 4.1).

mcpbrnn_Main_PETconstraint_Trial1.py: Single MCP node with time-variable output and loss gate plus regularization on evapotranspiration flux with mean/standard deviation of cell state updated through the results obtained in **mcpbrnn_Main_Trial1_Iter1.py** ($MC\{O_\sigma L_\sigma^{con}\}$ in session 4.2).

mcpbrnn_Main_PETconstraint_Trial2.py: Same as **mcpbrnn_Main_PETconstraint_Trial1.py** with mean/standard deviation of cell state updated ($MC\{O_\sigma L_\sigma^{con}\}$ in session 4.2).

mcpbrnn_Main_lossANNGate_PETconstraint.py: Single MCP node with time-variable output and loss gate plus regularization on evapotranspiration flux with increased functional complexity (ANN-based function) on loss gate ($MC\{O_{Ai} L_\sigma^{con}\}, 1 \leq i \leq 5$ in session 4.4.1).

mcpbrnn_Main_outputANNGate_PETconstraint.py: Single MCP node with time-variable output and loss gate plus regularization on evapotranspiration flux with increased functional complexity (ANN-based function) on output gate ($MC\{O_\sigma L_{Aj}^{con}\}, 1 \leq j \leq 5$ in session 4.4.1).

mcpbrnn_Main_ANNGate_PETconstraint.py: Single MCP node with time-variable output and loss gate plus regularization on evapotranspiration flux with increased functional complexity (ANN-based function) on both output gate and loss gate ($MC\{O_{Ai} L_{Aj}^{con}\}, 1 \leq i \leq 5; 1 \leq j \leq 5$ in session 4.4.1).

mcpbrnn_Main_MI_Loss_PETconstraint_Sigmoid.py: Single MCP node with time-variable output and loss gate plus regularization on evapotranspiration flux with increased context-dependence on loss gate ($MC\{O_\sigma L_{\sigma+}^{con}\}$ in session 4.4.2).

mcpbrnn_Main_MI_Output_PETconstraint_Sigmoid.py: Single MCP node with time-variable output and loss gate plus regularization on evapotranspiration flux with increased context-dependence on output gate ($MC\{O_{\sigma+} L_\sigma^{con}\}$ in session 4.4.2).

mcpbrnn_Main_MI_LossOutput_PETconstraint_Sigmoid.py: Single MCP node with time-variable output and loss gate plus regularization on evapotranspiration flux with increased context-dependence on output gate and loss gate ($MC\{O_{\sigma+} L_{\sigma+}^{con}\}$ in session 4.4.2).

mcpbrnn_Main_MI_Loss_PETconstraint.py: Single MCP node with time-variable output and loss gate plus regularization on evapotranspiration flux with increased context-dependence on loss gate using ANN-based function ($MC\{O_\sigma L_{Aj+}^{con}\}, 1 \leq j \leq 5$, session 4.4.2).

mcpbrnn_Main_MI_Output_PETconstraint.py: Single MCP node with time-variable output and loss gate plus regularization on evapotranspiration flux with increased context-dependence on output gate using ANN-based function ($MC\{O_{Ai+} L_\sigma^{con}\}, 1 \leq i \leq 5$, in session 4.4.2).

mcpbrnn_Main_MI_LossOutput_PETconstraint.py: Single MCP node with time-variable output and loss gate plus regularization on evapotranspiration flux with increased context-dependence on output gate and loss gate using ANN-based function ($MC\{O_{Ai+} L_{Aj+}^{con}\}, 1 \leq$ i, $j \leq 5$, in session 4.4.2).

MCA1-MR-Independent.py: Single MCP node with time-variable output and loss gate plus regularization on evapotranspiration flux with Cell-State-Independent Mass Relaxation gating ($MC\{O_\sigma L_\sigma^{con}\} M_I^R$, in session 4.5.1).

MCA1-MR-Independent_Iter1.py: Same as **MCA1-MR-Independent.py** with mean/standard Deviation of cell state value updated ($MC\{O_\sigma L_\sigma^{con}\}M_I^R$, session 4.5.1).

MCA1-MR-Independent_Relaxed.py: Single MCP node with time-variable output and loss gate plus regularization on evapotranspiration flux with Cell-State-Independent Mass Relaxation gating by allowing threshold value c_{MR} to be able to be positive or negative ($MC\{O_\sigma L_\sigma^{con}\}M_{Ir}^R$, session 4.5.1).

MCA1-MR-Independent_Relaxed_Iter1.py: Same as **MCA1-MR-Independent_Relaxed.py** with mean/standard Deviation of cell state value updated ($MC\{O_\sigma L_\sigma^{con}\}M_{Ir}^R$ in session 4.5.1).

MCA1-MR-Independent_Relaxed_Iter2.py: Same as **MCA1-MR-Independent_Relaxed.py** with mean/standard Deviation of cell state value updated from the result of **MCA1-MR-Independent_Relaxed_Iter1.py** ($MC\{O_\sigma L_\sigma^{con}\}M_{Ir}^R$ in session 4.5.1).

MCA1-MR-Regular.py: Single MCP node with time-variable output and loss gate plus regularization on evapotranspiration flux with Cell-State-dependent Mass Relaxation gating by allowing threshold value c_{MR} to be able to be positive or negative ($MC\{O_\sigma L_\sigma^{con}\}M_\sigma^R$ in session 4.5.1).

MCA1-MR-Regular_Iter1.py: Same as **MCA1-MR-Regular.py** with mean/standard Deviation of cell state value updated ($MC\{O_\sigma L_\sigma^{con}\}M_\sigma^R$ in session 4.5.1).

MCA1-MR-Regular_Relaxed.py: Single MCP node with time-variable output and loss gate plus regularization on evapotranspiration flux with Cell-State-dependent Mass Relaxation gating by allowing threshold value c_{MR} to be able to be positive or negative ($MC\{O_\sigma L_\sigma^{con}\}M_{\sigma r}^R$ in session 4.5.1).

MCA1-MR-Regular_Relaxed_Iter1.py: Same as **MCA1-MR-Regular_Relaxed.py** with mean/standard Deviation of cell state value updated ($MC\{O_\sigma L_\sigma^{con}\}M_{\sigma r}^R$ in session 4.5.1).

MCA1-MR-Regular_Relaxed_Iter2.py: Same as **MCA1-MR- Regular_Relaxed.py** with mean/standard Deviation of cell state value updated from the result of **MCA1-MR- Regular_Relaxed_Iter1.py** ($MC\{O_\sigma L_\sigma^{con}\}M_{\sigma r}^R$ in session 4.5.1).

mcpbrnn_Main_PETconstraint_IBcorrPL.py: Single MCP node with time-variable output and loss gate plus regularization on evapotranspiration flux with Piece-wise linear input-bias correction gate ($MC\{O_\sigma L_\sigma^{con}\}B_{Li}$, $1 \leq \mathrm{i} \leq 5$ in session 4.5.2).

mcpbrnn_Main_PETconstraint_IBcorrPQ.py: Single MCP node with time-variable output and loss gate plus regularization on evapotranspiration flux with Piece-wise quadratic input-bias correction gate ($MC\{O_\sigma L_\sigma^{con}\}B_{Qi}$, $1 \leq \mathrm{i} \leq 5$ in session 4.5.2).

mcpbrnn_Main_MI_LossOutput_PETconstraint_Sigmoid_MR_depedent.py: Single MCP node with time-variable output and loss gate plus regularization on evapotranspiration flux with increased context-dependence on output gate and loss gate and Cell-State-dependent Mass Relaxation gating ($MC\{O_{\sigma+} L_{\sigma+}^{con}\}M_\sigma^R$ in session 5).

mcpbrnn_Main_MI_LossOutput_PETconstraint_Sigmoid_MR_depedent_Iter1.py: Same as **mcpbrnn_Main_MI_LossOutput_PETconstraint_Sigmoid_MR_depedent.py** with mean/standard Deviation of cell state value updated ($MC\{O_{\sigma+} L_{\sigma+}^{con}\}M_\sigma^R$ *in* session 5).

We refer the readers to Table S1 in the supplementary material of Wang and Gupta (2023) for more information regarding parameter inheritance for each case as well as the updated mean/standard deviation of cell state.

d. Evaluation_Script: Each .py file in this folder is used to evaluate a specific case presented in this study.

The files in this folder were used to obtain the results for each case after training. The file names are kept the same as in **Training_Script** folder with "_EVAL" added at the end of each file name.

e. BM_Script: Each .py file in this folder is used to train and evaluate a specific benchmark case presented in this study.

TD-ARX_qsim.py: Training and evaluation for benchmark ARX case (session 5).

TD-ANN_qsim.py: Training and evaluation for benchmark ANN case (session 5).

Sigmoid-RNN-Vanilla_SingleNode.py: Training and evaluation for benchmark single node RNN case (session 5).

Sigmoid-RNN-Vanilla.py: Training and evaluation for benchmark multi-nodes RNN case (session 5).

LSTM-BM-singlenode.py: Training and evaluation for benchmark single node LSTM case (session 5).

LSTM-BM-multinode.py: Training and evaluation for benchmark multi-nodes LSTM case (session 5).

In short, the reason for providing .py file for each specific MCP case is because the parameter inheritance for each model is unique. There seems to be no unifying way of putting all the cases together.

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 the 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 MCP 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.

Finally, the user can train/evaluate the benchmark case by moving one of the training scripts from **BM_Script** folder back into the parent directory and then running 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.

4. Final Reminder

Notice that 1) the current version of the code is only able to run certain cases regarding the input variables used. For ARX, ANN, and RNN, only the case uses current-day precipitation, potential evapotranspiration, and simulated discharge (or cell state) at 1 previous timestep. 2) Also, the

code is designed to select the best epoch based on the highest KGE score for the *"selection set"*. The adjustment for model selection requires minor changes to be made in the script. 3) The number of epoch used in the training script might not be equal to the number listed in Table S1. Make sure to use "—epoch_no" parser to enter the correct epoch number when reproducing the results.