Response to CC1: 'Comment on egusphere-2024-2147', Chaopeng Shen, 22 Aug 2024

In this document we answer the questions and comments given by Chaopeng Shen. In summary, we consider that our implementation of the hybrid model and the nature of the experiments conducted in our experiment are appropriate and correct. Possible reasons for the differences with the results presented by Chaopeng Shen will be explained below. For clarity, we will answer each comment individually and explain why certain choices have been made in the preparation of the manuscript.

1. <u>Package difference</u>

Chaopeng indicates that:

> *Espinoza24 used their own training framework built on NeuralHydrology (NH) package including a sequence-to-one LSTM network, while we employed our framework (HydroDL) with a sequence-to-sequence LSTM network with slightly different implementations.*
>
> *While it seems the NH-hybrid package can give the same performance in the benchmark case used in Feng et al 2022, it gives suboptimal performance in extreme event tests.*

This is not correct. Our hybrid model implementation in NH is also trained using sequence-to-sequence in the same manner has it has been implemented in Feng et al. (2022).

2. <u>Ensemble strategy</u>

Chaopeng indicates that:

> Espinoza24 used ensemble LSTM to compare with a single dHBV model
>
> *Espinoza24 misinterpreted the dHBV hybrid model. The multicomponent in HBV is not an "ensemble". Rather, it is similar to the hidden size in LSTM. The real ensemble of differentiable models should be composed of different model structures, e.g., HBV, SAC-SMA, PRMS. We have ongoing work that shows the true model ensemble provides better NSE. That being said, the impact on extreme impact needs more time to understand.*

We do not agree with this argument. There is no scientific reason why an ensemble should consist of different model structures. There are multiple ways in which an ensemble can be accomplished, for example: training multiple models with different initializations (like we did with the stand-alone LSTM), propagating the uncertainty of the inputs throughout the model to get a probabilistic output, using multiple models in parallel and weighting their outputs, among other strategies. E.g. in operational weather forecast practice, ensemble forecasts are mainly created by perturbed initial conditions.

Therefore, there is no reason why *a "real ensemble of differentiable models should be composed of different model structures"*. Chaopeng Shen indicates that they have done work in which different model structures give an advantage, and we do not doubt this is the case. It is perfectly feasible that different model structures do increase the performance. However, this does not indicate that using the same structure is not considered a "real ensemble". Moreover, their work on hybrid model ensembles was not published at the moment we ran the experiments, so we could not use this information in this study.

3. Experiment design

Chaopeng Shen indicates that:

> *Espinoza24 et al. validated the model using holdout years within the training period--- they trained the model with CAMELS data from 1980-2014, and tested it using years within 1980-2014 that had extreme events and were held out during training.*
>
> *We trained from 1995/10/01 to 2014/09/30, holding out water years with peak flows greater than a 5-year return period. We then tested models on a separate continuous time period, from 1980/10/01 to 1995/09/30.*
>
> *We argue a true test should purely exist in continuous history or future years to avoid any kind of information leak. Even though Espinoza24 would say the data in some years is held out for test, during training LSTM still sees what the future looks like after the extreme events. Somehow this makes it a simpler task than purely predicting in untrained years. We have run the tests, a pure extrapolation like what we did represents a harder case, and LSTM shows more disadvantages less favored under such a more real-world scenario.*
>
> *Further, we observed that alternative choices (Table C1) in experimental design, which might align more closely with Frame et al., 2022, or represent more realistic tests, could yield greater advantages for dHBV HydroDL over LSTM (Figure C3). Specifically, the experiments in Espinoza24 may allow LSTM to operate more within an "interpolation" regime, whereas true experiments should challenge models in an "extrapolation" context.*

We do not agree with this argument. It is not true that "*a true test should purely exist in continuous history or future years to avoid any kind of information leak. Even though Espinoza24 would say the data in some years is held out for test, during training LSTM still sees what the future looks like after the extreme events. Somehow this makes it a simpler task than purely predicting in untrained years*"

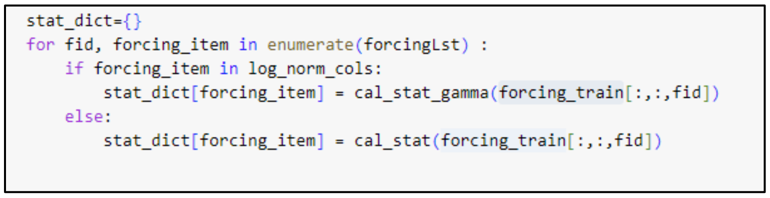
We included a buffer period of one year between the training and testing years, therefore the LSTM does not see" what the future looks like after the extreme events", nor is there any kind of information leaking from testing to training. The training/testing split was done based on the probability of the data, giving different conditions and presenting unseen events to the model during testing. Illustrating this idea with an example, we do not see any reason why, if a model was trained using low flow years in 1999 and 2003, this would give an advantage when predicting an extreme event in 2001.

Possible reasons for the differences in performance

Looking at the code uploaded by the authors, we found two possible reasons for the differences in the results.

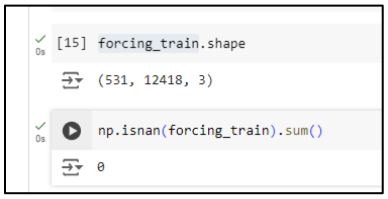
Data standardization

In the code provided by Chaopeng Shen (https://colab.research.google.com/drive/12xUvTu9NoGVdcRWqJvypy4DGg5jy5q9T#scrollTo=v%200JIEuFZjkxq), the statistics to standardize the input data that goes into the LSTM are calculated as follows:



```
stat_dict={}
for fid, forcing_item in enumerate(forcingLst) :
    if forcing_item in log_norm_cols:
        stat_dict[forcing_item] = cal_stat_gamma(forcing_train[:,:,fid])
    else:
        stat_dict[forcing_item] = cal_stat(forcing_train[:,:,fid])
```

The numpy array *forcing_train* has dimensions (531, 12418, 3), and there are not any NaN values.

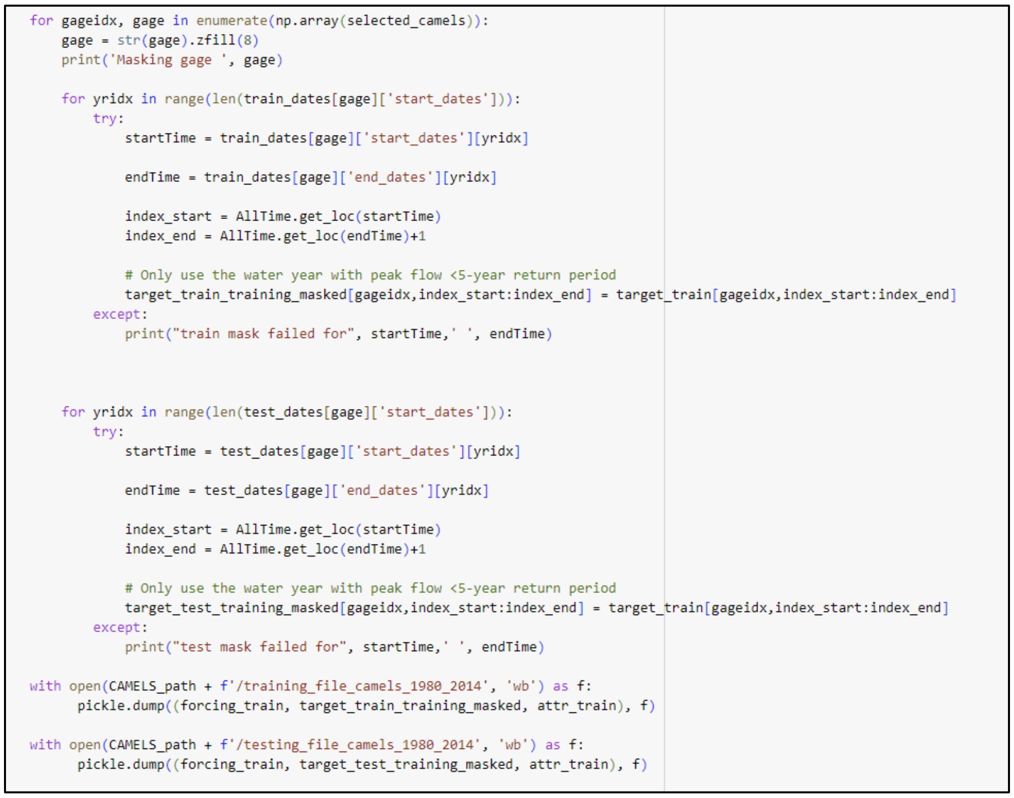


```
[15] forcing_train.shape

(531, 12418, 3)

np.isnan(forcing_train).sum()

0
```

The second dimension indicates that one has 12 418 days (period between 1980-2014). Therefore, the authors are calculating the mean and standard deviation used to standardize the input data using the whole period (training and testing). In our case, we calculated the statistics using only the training years, to avoid information leaking. We believe this might be one of several other reasons for the different results.

Training strategy

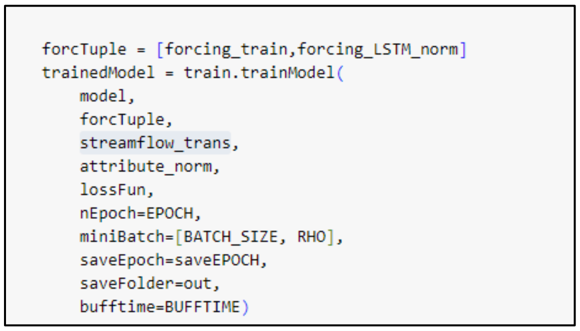
In the code provided by Chaopeng Shen, the discharges are masked depending if one is in training/testing mode. Therefore, no testing discharge information is used to calculate the loss during training.



```
for gageidx, gage in enumerate(np.array(selected_camels)):
    gage = str(gage).zfill(8)
    print('Masking gage ', gage)

    for yridx in range(len(train_dates[gage]['start_dates'])):
        try:
            startTime = train_dates[gage]['start_dates'][yridx]

            endTime = train_dates[gage]['end_dates'][yridx]

            index_start = AllTime.get_loc(startTime)
            index_end = AllTime.get_loc(endTime)+1

            # Only use the water year with peak flow <5-year return period
            target_train_training_masked[gageidx,index_start:index_end] = target_train[gageidx,index_start:index_end]
        except:
            print("train mask failed for", startTime,' ', endTime)


    for yridx in range(len(test_dates[gage]['start_dates'])):
        try:
            startTime = test_dates[gage]['start_dates'][yridx]

            endTime = test_dates[gage]['end_dates'][yridx]

            index_start = AllTime.get_loc(startTime)
            index_end = AllTime.get_loc(endTime)+1

            # Only use the water year with peak flow <5-year return period
            target_test_training_masked[gageidx,index_start:index_end] = target_train[gageidx,index_start:index_end]
        except:
            print("test mask failed for", startTime,' ', endTime)

with open(CAMELS_path + f'/training_file_camels_1980_2014', 'wb') as f:
      pickle.dump((forcing_train, target_train_training_masked, attr_train), f)

with open(CAMELS_path + f'/testing_file_camels_1980_2014', 'wb') as f:
      pickle.dump((forcing_train, target_test_training_masked, attr_train), f)
```

However, their training strategy does differ from ours and it also differs from how they are training the stand-alone LSTM (given that they are using Neural Hydrology for the last one).

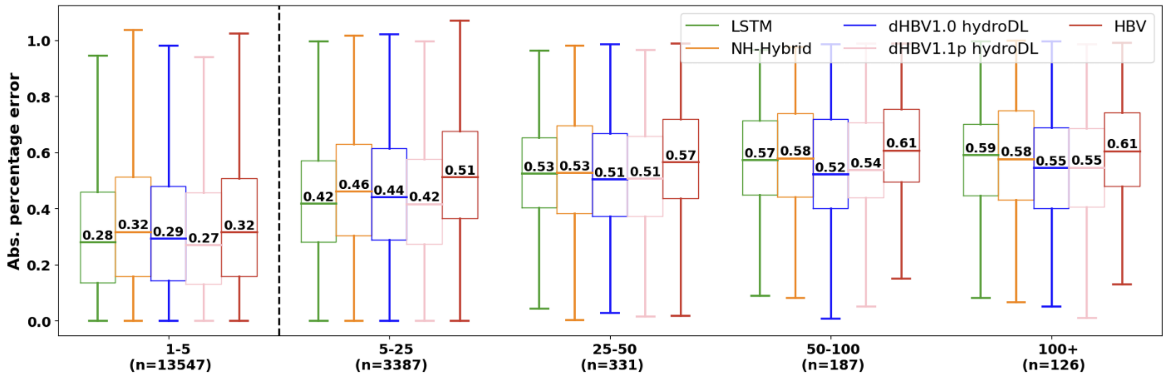
During training, they constructed the batches using information from the whole period (1980-2014), which they send to the model in the *forcTuple* list. This includes both training and testing years. Therefore, during training, for some elements of the batch, the model does a forward pass of information contained in the testing regime. The associated simulated values are not used to calculate the loss during the optimization; however, this strategy is indeed different from the strategy we used.


```
forcTuple = [forcing_train,forcing_LSTM_norm]
trainedModel = train.trainModel(
    model,
    forcTuple,
    streamflow_trans,
    attribute_norm,
    lossFun,
    nEpoch=EPOCH,
    miniBatch=[BATCH_SIZE, RHO],
    saveEpoch=saveEPOCH,
    saveFolder=out,
    bufftime=BUFFTIME)
```

For our hybrid model and for the stand-alone LSTM, the batches that we used during training do not contain any values located in the testing period, neither inputs nor targets, and during training no forward pass of any testing value is done. We believe that this consistency in training both models in the similar manner is beneficial.

<u>Reported differences</u>

Lastly, we would like to highlight that in 4 out of the 5 cases we evaluated, the differences in the models do not change the conclusions we showed. In Figure C1 uploaded by Chaopeng Shen (which we copy below for clarity) we can see the following:



Source: https://doi.org/10.5194/egusphere-2024-2147-CC1

a. In the 1-5 and 5-25, the ranking between the LSTM with NH-Hybrid and LSTM with dHBV1.0 stays the same.
b. For the 25-50 and 100+, we did not indicate in our manuscript that the LSTM performed better. We indicated that both models performed similarly, and the differences can be

explained by statistical noise. This is still the case with dHBV1.0. As we showed in Figure B1 of our manuscript (see figure below), different random initializations of the LSTM can create variation in the reported metric. We can see that for the second row of the Figure below, the LSTM can achieve a median values of 0.51 and 0.56 for the 25-50 and 100+ intervals, which are close to the 0.51 and 0.55 reported by the dHBV1.0 for these same cases.

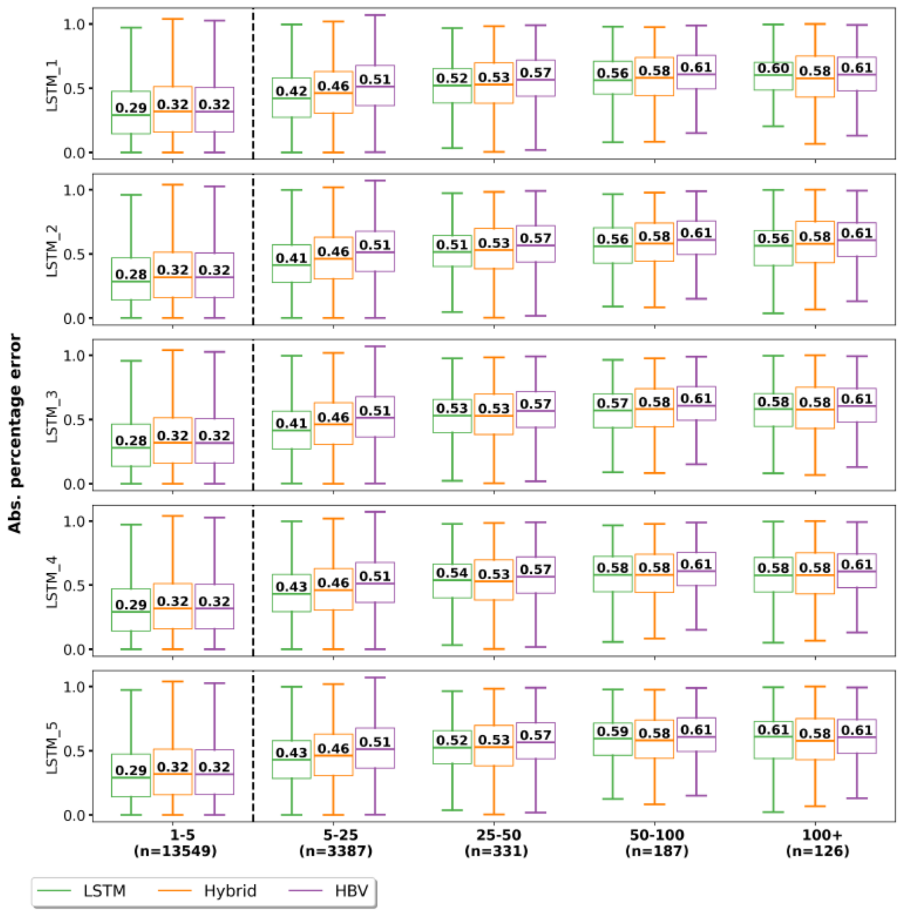


Figure B1. Variation in absolute percentage error due to random initialization of the LSTM model.

c. In the 50-100 interval, it does looks like there is a difference between both implementations of the hybrid model (NH-Hybrid and dHBV1.0), that we believe can be explained by the reasons stated in the previous sections.

Sincerely,

Eduardo Acuna Espinoza on behalf of all co-authors