The authors replied:

> 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.

To test the authors' argument that this was statistical noise, we further run more random seeds on the dHBV1.0 (dHBV1.1p was not retrained with more random seeds as we don't have enough time before the comment session closes), as shown in Figure CC1 ("hybrid" is what Espinoza24 trained while dHBV1.0 are ours). It turns out the authors' argument was not correct. In all of these random seeds, we see a steady outperformance of dHBV1.0 over LSTM for 25-50, 50-100 and of course 100+ cases. In fact we can certainly run a statistical analysis to verify the statistical significance with more random seeds. LSTM is better in the 5-25 than dHBV1.0 but about the same as dHBV1.1p (only one random seed). That case precisely shows that LSTM is better at cases close to what is has seen in training, and worse for those cases that it has not seen.

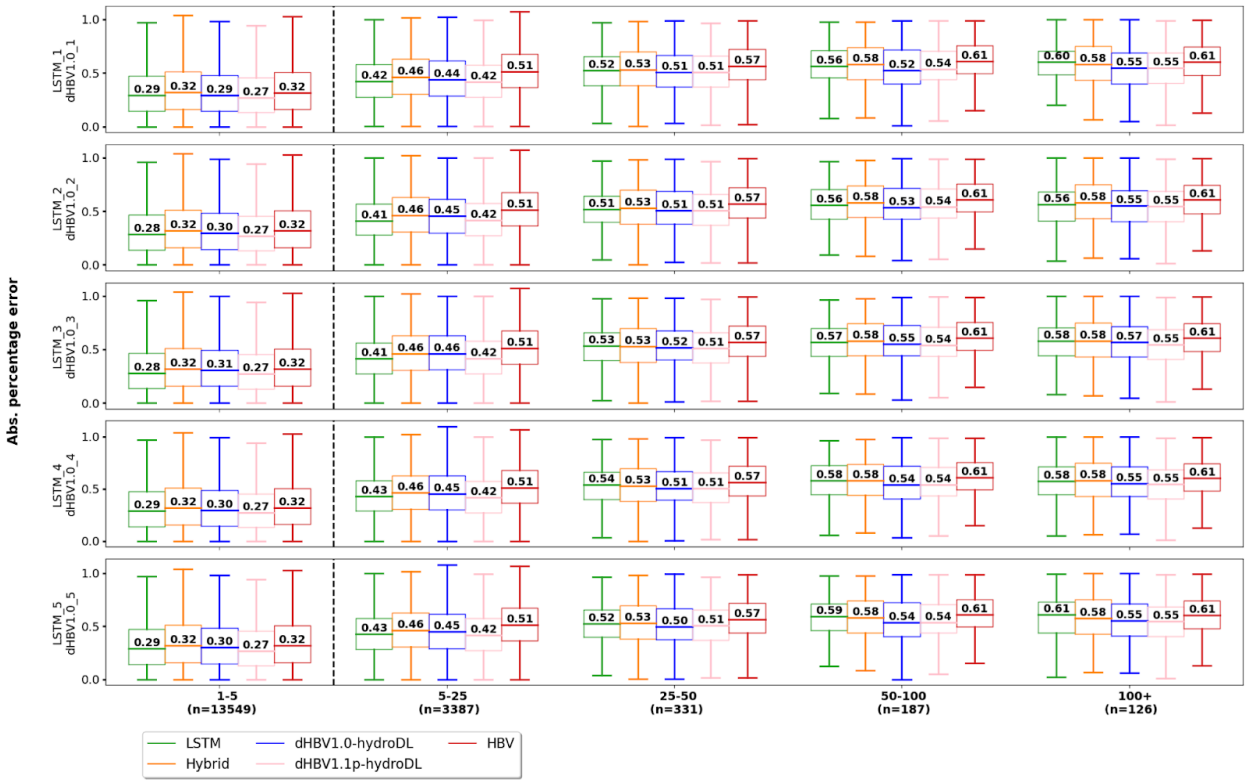


Figure CC1. We re-ran the experiment with more random seeds. "hybrid" is what Espinoza24 trained while dHBV1.0 & dHBV1.1p are trained by us. Each row is the result from a random seed. The random seeds used for dHBV1.0 were 111111, 22222, 33333, 44444, 55555. The 1.1p was trained using the same random seed due to time limitation.

It seems fair to say the “hybrid” model trained by the authors is not representative of the dHBV1.0 as in every random seed the dHBV1.0 had smaller errors --- there is not even one exception. The difference between them is due to the different training frameworks employed, as we explained in the first comment. We leave it to other readers to interpret the differences, but, from our reading, the authors “hybrid” would suggest LSTM tend to outperform while our figure would suggest dHBV tend to outperform for the extremes. It seems fair to say the community would be better served by involving at least dHBV1.0-hydroDL into the comparison to draw a more balanced conclusion accordingly.

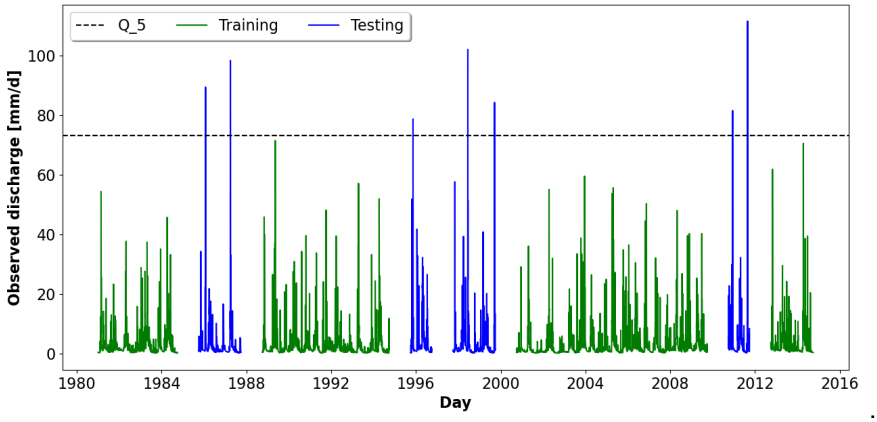
The point about input scaler

The authors further argued that some minor differences in the setups caused the difference. The authors replied:

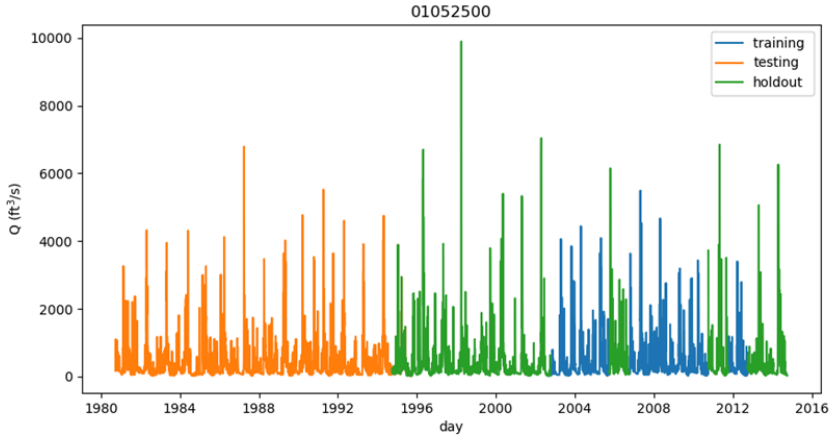
> The authors (us) are calculating the mean and standard deviation used to standardize the input data using the whole period (training and testing). In our case, we calculated the statistics using only the training years, to avoid information leaking. We believe this might be one of several other reasons for the different results.
>
> 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.

First, this was a minor scaler setup and there was no data leakage because precipitation as an input is supposedly known or can be assumed for the purpose of calculating the scaler. Second, we proposed an experiment which cleanly and easily separates out training and test (the temporal extrapolation case shown in our first comment) where the advantages of dHBV were more prominent in our test and we again encourage the authors to run a case like that with NH. At least, through the temporal extrapolation case we can see that the issue mentioned by the author does not have a noticeable impact. Third, one can make some further effort to have a cleaner scaler. We have taken 99 steps to get close to their exact setup and trust the authors can bridge the last 1 step.

In reality, we don’t encounter scenarios where we know both the historical and future time series and test in the middle of the time series. How we use the model is like what is shown in Figure CC2b. In this experimental design, the model is trained on water years with lower return periods (blue line), while water years with higher return periods (green line) are held out from training. The model is then tested over a separate time span that includes both extreme and low flow events.



(a)



(b)

Figure CC2. Different experimental designs.

The final point about ensemble.

We still think it is unfair to compare ensemble LSTM with a single dHBV. The multicomponent in dHBV is like the hidden size in LSTM. Here is a simple criterion: an ensemble of n LSTM has n neural networks, whereas a dHBV has only one neural network. Because of the constraint imposed by HBV, it is not as random as LSTM so random seeds are not what one should do to get an ensemble for dHBV. More effort will be shown down the road on this topic.

Overall, this point is not highly relevant to the extreme discussion, here, now. Nevertheless, comparing ensemble LSTM with a single dHBV still feels like “bringing everything you’ve got” on the LSTM side while not doing much on the dHBV side.

Finally, we would like to say that whether this paper gets published or not is not our concern --- we just want to ensure the community gets the full picture and get a balanced view.