# Comment on “Towards Interpretable LSTM-based Modelling of Hydrological Systems”

## Summary of Paper:

The objective of this paper is to allow for interpretability of the weights of an LSTM model for streamflow prediction. The authors approach this objective by making two changes to their baseline model (from Line 200 in the manuscript):

1) They use sequence-to-sequence prediction
2) They give each cell of the LSTM a sequence of lagged input data.

## Comment #1:

I believe that there is a problem with one of the fundamental arguments that appears throughout the paper, related to the relationship between the model and the physical system.

- Line 10: *“Our architecture, called HydroLSTM, simulates behaviors inherent in a dynamic system, such as sequential updating of the Markovian storage”*
- Line 538: *“We have proposed and tested a more interpretable LSTM architecture that better aligns with how we conceptualize the physical functioning of hydrologic systems.”*

What the authors did is actually break the Markovian nature of the LSTM. The regular LSTM is Markovian, in that the prediction at time t is dependent only on the state of the system, and the inputs at time t. By adding lagged inputs to each LSTM cell, the authors have made the model non-Markovian. This explicitly breaks the isomorphism between the LSTM and the physical system. The physical system is Markovian, since the watershed cannot “see” yesterday’s rainfall, except through the effect on the various storage states within the system. The addition of lagged inputs actually aligns significantly *worse* with how we conceptualize physical hydrological systems.

Moreover, the authors claim (Line 45) that 45: *“It is possible, therefore, that either the corresponding LSTM-based models are not efficient (parsimonious) representations of the input-state-output dynamics, or that our conceptual hydrological models are overly simplified representations of reality (over-compression). In this paper, we make an argument for the former explanation.”* I do not believe that this is a correct interpretation of the experiments in this paper. What the authors have done is to show that if you remove the Markovian assumption from the model, then you can replace information that in the physical system would be stored as a state variable with (non-markovian) lagged input data, which the real system does not have access to (soil can't "see" rainfall from yesterday except through the current state of the watershed). I think you've just shown that non-Markovian models require less memory than Markovian models, which, in my opinion, is obvious.

It is the case that we do know that the LSTMs do not necessarily optimize to reduce the number of states. You can see this by doing an experiment where you emulate (exactly reproduce) a conceptual model - the LSTM typically requires more states than the "true" system (as defined by the conceptual model). This is a byproduct of inefficient (local) optimization, which is all that can be achieved using backpropagation. However, I don't think that the experiments in this paper demonstrate the claim, since you've allowed the number of states to be reduced due to breaking the memory requirement of the model - of course if you show the model yesterday's precipitation in today's state/output calculation, then the model doesn't need to remember as much information.

## Comment #2:

I believe there is a misunderstanding about sequence-to-sequence (seq2seq) vs sequence-to-one (seq2one) prediction.

The authors state in line 150 *"In these areas, two primary assumptions are typically applied that may not hold in the dynamic environmental system: a) a finite relevant sequence length (finite memory time-scale), and the consequent possibility of 150 b) a non-informative system state initialization."*

They also state in line 200 "*the cell states are continually updated from the beginning to the end of the available dataset while maintaining the sequential ordering of the input drivers. This ensures that the cell states represent Markovian memories that are effectively of indeterminate length (as in traditional hydrological modeling, initialization is done only once at the beginning of the simulation period)"*

What the authors are describing is the difference between seq2seq vs. seq2one. The Kratzert papers use seq2one for *training*, but seq2seq for *prediction*. The reason that seq2seq is used for training is to help randomize the minibatch. Seq2seq allows for dividing the total training sample up so that any single minibatch contains samples from multiple different watersheds. This helps prevent spurious weight updates, which can significantly harm training. However, once the models are trained, they are applied as these authors describe — simulating the entire sequence without restarting (i.e., seq2seq prediction). The authors might notice that this is why the Krazert models require O(10) epochs for training while their seq2seq approach requires O(100) epochs for training.

It's also important to point out that this is not actually a difference in the model itself, but instead a difference in how the model is applied (seq2seq models are identical to seq2one models, just applied to different types of data). This is just a distinction in terms of how this is framed in the discussion (the authors refer to this in line 199 as a difference in model structure, which it is not).

The authors should be aware that this seq2seq vs. seq2one distinction does not actually make a difference (other than the minibatch diversity thing during training, which is important). The reason why it does not matter is because of the forget gate. The forget gate can be thought of as a repeated multiplication operation. If the forget gates are not exactly one, then the memory of the model is limited by this repeated operation. Imagine repeatedly applying the forget gate operation on information in cell states from 300 time steps ago. If the forget gate has values close to one (i.e., fully open, meaning that they try to forget as little information as possible), then you are applying a repeated operation of multiplying by a number close to one several hundred times. As an example, 0.99 ^ 300 = 0.05, meaning that even in a very optimistic case where the forget gate is almost completely open at all timesteps, only 5% of the information is left after 300 timesteps *purely because of numerical artifacts in the asymptotes of the forget gate activation functions.* It does not make very much sense to extend the input time sequence very much beyond this, simply because there will be no mathematical effect. This is why we are comfortable using seq2one for training, and the reason we use seq2seq for inference is because it's simply not necessary to do the extra computations necessary for seq2one when there is no minibatch.

I'd encourage the authors to test this – how much memory can the model have before it loses all sensitivity to inputs in the distant past? This can be answered using integrated gradients (we have done this experiment). Notice that if the authors wanted to avoid this numerical artifact, they could input a very long sequence directly into each cell state of the LSTM, however the number of weights in the model would explode. At some point, you might as well just use a time convolution (or similar), and not worry about the cell states at all.

## Comment #3:

The authors cited my 2021 paper in the first paragraph of the introduction, however that paper does not say anything similar to what is claimed in the sentence where it is cited (and I strongly disagree with the opinion expressed in that sentence). I would kindly ask the authors to please either remove this incorrect citation or else modify it to more accurately reflect what is written in the paper they are citing.

## Comment #4:

It would be very helpful if the authors would benchmark against an existing published paper. The authors have changed the set of basins that they train and test on (Line , as well as the train and test periods. This means that there is no way to know whether the authors have set up their LSTM in a way that matches the current state of the art. Their results are not directly comparable to anything that has previously been published. This should be easy, since the authors are using the same CAMELS dataset that numerous previous authors have used to develop and test the baseline ML model used in this study. It is OK to make changes to the train and test settings, but it would be very helpful if the authors provided a community benchmark to help us know whether to trust their results. The authors might see [1] for and example where we felt that it was necessary to change the train/test periods for a specific experiment, but also published benchmarks against previous publications in the same paper, to ensure readers that our models were performing near state of the art.

## Comment #5:

It is important to note that the integrated gradients method that the authors discuss in the introduction actually provides effectively the same information that this new method provides. Both methods give us relative importances on different model input channels. The authors could see [2] for an example of using integrated gradients to understand sensitivity of an LSTM to lagged input data. So what is this new method supposed to help us with? What can be learned from this that cannot be learned by using a method that does break the isomorphism between the model and the physical system (by making the model non-Markovian), and that does not harm the performance of the model?

After reading this paper, my main question for the authors is this: What problem are you trying to solve? What do you want to be able to do that can’t already be done (and hasn’t already been done)? Is there some limitation to what we can learn using explainability methods, and if so, what are those limitations?

## References:

[1] Frame, J. M., Kratzert, F., Klotz, D., Gauch, M., Shalev, G., Gilon, O., Qualls, L. M., Gupta, H. V., and Nearing, G. S.: Deep learning rainfall–runoff predictions of extreme events, Hydrol. Earth Syst. Sci., 26, 3377–3392, https://doi.org/10.5194/hess-26-3377-2022, 2022.

[2] Kratzert, F., Klotz, D., Hochreiter, S., and Nearing, G. S.: A note on leveraging synergy in multiple meteorological data sets with deep learning for rainfall–runoff modeling, Hydrol. Earth Syst. Sci., 25, 2685–2703, https://doi.org/10.5194/hess-25-2685-2021, 2021.