Response to Grey Nearing’s comments.

Firstly, we would like to thank you for reviewing and commenting on our paper. We find this discussion very interesting and feel that it will enrich the final version of the paper.

Reviewer 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.

Author Response:

Our interpretation of a Markovian process is that, for each time step, ***all*** of the information provided by the past data that is useful for making the next-time-steps prediction is contained in the system “states” (however those are determined). Based on this, one might argue that *neither* the LSTM nor the HydroLSTM is fully “Markovian”.

The conventional LSTM implementation assumes some arbitrary initial value for the cell-states at some assumed/fixed lagged time in the past and then processes some past data sequentially (in a Markovian manner) to develop an estimate of important properties of the system state at time t (when a prediction for the next time step is desired). The assumption, therefore, is that there is no useful information (relevant to making that prediction) in the data representing times before that assumed sequence length. Arguably, this goes against hydrological knowledge (whether correct or erroneous), which suggests that process memory can exceed 365 days, which is a very common sequence length used in LSTM-based hydrological modeling). Under such conditions, the conventional LSTM-based representation is not able to exploit all of the information that should otherwise be summarized in the representation of the system state for use in prediction, which arguably also breaks the Markovian assumption.

Similarly, the proposed HydroLSTM architecture also uses sequential updating “akin” to a Markovian process to estimate important properties of the system state at time t, for use in making the prediction for the next time step. And as with the LSTM, the gating functions define how the “cell-state” updating is done. However, because the gating functions use past lagged data to establish context for such gating, the gates have access to more information than is provided by the current forcing and cell-states. As you have pointed out, this also breaks the traditional Markovian nature of the cell-state representation and updating process, because it allows for important information regarding the overall “state of *the system*” to be explicitly provided via sequence lagged data fed to the gating functions to be used in determining how open or closed they are. In other words, the “cell-states” now do not represent the entire “*state*” of the system because they no longer contain *all* of the information provided by the past data that is useful for making the next-time-steps prediction. Arguably the true “system state” is now represented by a combination of the “*cell-states*” *and* the sequence information provided to the gates, since both are used to inform the next time-steps prediction.

Which way of breaking the classical “Markovian” behavior is better or worse is arguably dependent on our goal. And it is not clear that cell-state summaries determined primarily from “Markovian-like” processing of the mass and energy-related time-series provided as inputs to the model are necessarily sufficient to characterize the true “state” of the system at any point in time. Therefore, our view is that whether or not a classical “Markovian” or "non-Markovian" (or even "partially Markovian") representation is useful for gaining insight into the internal relationships learned by a Machine Learning model is not the most important thing.

Regarding the comment that the LSTM representation does not necessarily optimize to reduce the number of states, this is an interesting issue that would be useful to develop a better understanding of. It could also, perhaps, be a consequence of the limitations imposed by the specific architectural choices available to the LSTM (including, for example, the forms of the activation functions used). And, as you mention in the next comments, it could also be associated with the non-conservative behavior of the forget gate. We are of the opinion that more investigation should be done to explore improvements or modifications to the current LSTM representation, at least when applied in a hydrological context.

In summary, we thank you for raising the discussion about the Markovian nature of the process, as it is a topic that will enrich the paper – as a consequence we plan to add a small discussion to the revision. Moreover, to avoid misunderstandings with the concept of Markovian we will use “quotes” when using this concept in reference to the proposed HydroLSTM architecture.

Reviewer 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.

Author Response:

We agreed that your description of seq2seq and seq2one sounds like what we described. However, we are trying to describe something slightly different. In hydrological models, a warm-up period is commonly used to establish the initial value of the state variable. After this period, the state variable is updated with each new piece of information (in our case, daily precipitation, and temperature). The warm-up period can be thought of as a seq2one process, while the posterior updating of the state resembles a seq2seq process. However, during the training of a seq2one model, this warming-up process is repeated for each time value in the time series, which can result in the model learning an average sequence length that does not consider anomalies with respect to the mean. To address this issue, the state value could be fed at the beginning of each training period, but this approach would not allow for the randomization of samples described in the comment. The process of beginning from zero constantly to define the state variable is what we describe as a process that is not like what we do when we solve a dynamic system.

The mention of seq2seq in the paper was to try and express that more ideas can be built up over LSTM than to mention it as another architecture. We will clarify this in the next version of the paper.
As you mentioned very well, the forget gate is non-conservative (asymptotic to 1) which truncates the past information until it is insensitive. Therefore, new mechanisms should be explored to deal with that. The idea of using time convolution is well-taken because HydroLSTM is basically using a time convolution inside of the gates. What we have found is that such filters can be informative, in an interesting way, about the hydrological relationships built inside the representation, which is the final goal of the paper.

Reviewer 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.

Author Response:

We apologize for this inaccuracy. The reference will be deleted in the revised version.

Reviewer 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.

Author Response:

We are not sure what you mean by changing the basin of training and testing, which probably means we should improve the description of our experiments in the next version.

We conducted two experiments. In the first experiment, we compared both representations using only 10 catchments, training one model per catchment. Each catchment was calibrated using data from January 1, 1980 to September 30, 2000. We then used the next four years to select the best epoch and the final period to present the results.

From the results of this experiment, we found that a single HydroLSTM cell had reasonably "good" performance compared to the best possible configuration of lag data and the number of cells. Therefore, in the second experiment, we explored what we could learn from this simplified representation. We independently trained one HydroLSTM cell for each of the 588 selected catchments (including the previous 10), keeping the same procedure as in the first experiment (i.e., splitting the data).

Please note, as was mentioned in the paper, that our goal is *not* to demonstrate that HydroLSTM has a better performance than the LSTM. It is highly probable that state of the art in LSTM representations and those that use of more cells than we used, could beat HydroLSTM in terms of performance. Instead, our aim was to explore the possibility of parsimonious representations (in terms of numbers of cell-states) with the goal of gaining insights into the interpretability of parameters and state variables. Given this purpose, we thought it was most important to select a subset of basins that represent the range of hydrologic behaviors we want to capture, rather than selecting basins where we could compare to previous studies.

Reviewer 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?

Response:

We agree that the gradient method effectively provides the same information. However, the HydroLSTM uses that information explicitly in the gates. This allows the user to more easily understand and visually interpret the "feature importance" encoded in the model. For that reason, the title of our paper is focused on interpretability, rather than on presenting HydroLSTM as any kind of new state of the art. Given that some people are still reluctant to use machine learning methods, because they are seeking more than just predictive performance, we feel that finding ways to make machine learning methods as interpretable as possible is a valuable goal.