Response to reviewer 1: Jens Kiesel

Link to review

Comments/Text of reviewer posted in **black**; our answers are posted in **blue**.

1. GENERAL COMMENTS

The manuscript "Rainfall–Runoff Prediction at Multiple Timescales with a Single Long Short-Term Memory Network" (LSTM) by Martin Gauch et al. presents an extension of LSTM hydrological models to sub-daily time steps. In previous publications, LSTMs as hydrological models were used on a daily time step. The authors explore multiple approaches to achieve a 'multi-timescale' model, of which three (naive LSTM, sMTS LSTM, MTS-LSTM) are evaluated in more detail and less promising experiments are briefly explained in an Annexe. Similar to previous applications of LSTMs, the models are applied at the CAMELS dataset, encompassing 516 basins across the contiguous USA, where hourly data is available. Results are compared to the NOAA National Water Model (NWM) and show that all LSTMs architectures outperform the NWM. The authors suggest that the MTS-LSTM provides most flexibility for future use.

The manuscript is generally well written and structured, figures and tables support the results. Having a more process-based hydrological background, I nevertheless read the paper with interest and believe it fits well in the scope of HESS. I see the work as highly relevant, especially in the field of flood modelling and (eventually) forecasting, but also generally in the application of LSTMs at different temporal resolutions. However, especially regarding the latter, I think the authors should invest more work to improve the usefulness of the paper. Please find below more detailed comments, questions and suggestions that hopefully initiate a fruitful discussion and help in improving the paper.

We would like to thank Jens Kiesel for his detailed and thoughtful review. Based on his comments, we have prepared a revised manuscript that (among other changes) provides a more accessible review of related work in machine learning and gives a more detailed description of the differences between our proposed multi-timescale LSTM variants. In the following, we address each comment individually. Our answers are colored in blue.

2. SPECIFIC COMMENTS

2.1. ABSTRACT

I suggest to mention the difficulties and challenges applying the models (parameter estimation) and discuss the work still to be done regarding different time scales (e.g. generalization of parameters)

2.2. INTRODUCTION

I think you are missing a research gap in your introduction which is important to apply the LSTM for different time steps, since there seems to be a time step dependency of model parameters / hyperparameters (e.g. hidden size, sequence length, batch size, forget gate bias, learning rate?, others?). Due to the computationally expensive training of LSTMs, knowing which ones need to be adjusted, in about which range and identifying ideal values is essential. I would like to see this topic included in the "contributions" you list at the end of the introduction (and therefore also more prominently in the respective chapters).

2.2.1. p.2 l.29-41: I think this section is difficult to understand for a reader without firm neural networks background. Particularly phrases like: "partitions a recurrent neural network into layers with individual clock speeds", "process irregularly sampled inputs by means of a time gate that only attends to the input at steps of a learned frequency", "the approach depends on a binary decision that is only differentiable through a workaround". I acknowledge that your paper cannot serve as an introduction to the topic. I have no clear suggestion other than making this paragraph more accessible to readers with a hydrological background through using less specialized jargon, if possible.

Thank you for the feedback. We agree with this analysis of the language. In the revised version, we will explain the related work and its connections to MTS-LSTM in terms that are more familiar to the intended audience as follows (formerly L29-41):

“More recently, Koutnik et al. (2014) proposed an architecture that efficiently learns long- and short-term relationships in time series. They partition the internals of their neural network into different groups, where each group is updated on individual, pre-defined, intervals. However, the whole sequence is still processed on the highest frequency, which makes training slow. Chung et al. (2016) extended the idea of hierarchically processing the different timescales: In their setup, the model can adjust its updating rates to the current input, e.g., to align with words or handwriting strokes. Unfortunately, the binary decision whether to make an update complicates the procedure, since it can only be trained through a workaround. Neil at al. (2016) proposed an LSTM architecture with a gating mechanism that allows state and output updates only during time slots of learned frequencies. For time steps where the gate is closed, the old state and output are reused. This helps discriminate superimposed input signals, but is likely unsuited for rainfall--runoff prediction because no aggregation takes place while the time gate is closed. In a different research thread, Graves et al. (2007) proposed a multidimensional variant of LSTMs for problems with more than one temporal or spatial dimension. Framed in the context of multi-timescale prediction, one could define each timescale as one temporal dimension and process the inputs at all timescales simultaneously. Like the hierarchical approaches, however, this paradigm would process the full time series at all timescales and thus lead to slow training."

2.2.2. p.2 l.45 and 47: You write that Araya et al. predicted wind speed at "multiple timescales". Then you mention that your objective is "multiple outputs, one for each target timescale". I don't understand the difference between that.

Given our current formulation it is understandable that the difference was unclear. By "they predicted wind speed given input data at multiple timescales," we meant that the *input* data to their model is at multiple timescales (e.g., hourly and daily input values). The *output*, however, is only at the hourly timescale. MTS-LSTM, in contrast, produces a distinct output for each timescale (e.g., discharge prediction at hourly and daily timescale). We will rephrase the sentence to clarify this.

2.2.3. p.2 l.54ff: I see the capability to process input data in irregular intervals as an advantage. Think of satellite products that have different data gap length (e.g. soil moisture or altimetry products combining multiple sensors). You can discuss this further, but at least I suggest to write on p.3 l.77: "...LSTM can ingest individual and multiple sets of forcings each having regular time intervals for each target timescale. This closely resembles..."

Thank you, we will adopt the suggestion.

2.2.4. p.3 l.70-72, 74-75: I suggest not to mention the results of your study in the introduction

We understand your suggestion. However, we believe that a brief description of the overall results in the introduction helps readers navigate the manuscript and emphasizes why the contributions are meaningful. Also, it serves people who only read the introduction/contributions and conclusion for a high-level overview on the paper. That said, if our view is strongly opposed by the editor or reviewers, we are open to revise the introduction accordingly.

2.2.5. p.3 l.64-78: These three paragraphs reveal that your introduction could be structured a bit better, ideally introducing the reader to these three problems/research gaps that need to be solved for "Rainfall–Runoff Prediction at Multiple Timescales with a Single Long Short-Term Memory Network". You have motivated the first paragraph, but the second and third 'contribution' that you list appears a bit unexpected since your previous introduction does not resemble that structure. For instance, instead of referring to sections later in the paper, I believe it would be better to introduce the reader to the problem of inconsistencies. You briefly mention this on p.2 l.27-28 for conventional hydrological models, but this can be extended, especially targeted on machine learning.

Thank you for these suggestions. In the revised manuscript, we will introduce the problems of inconsistency and of per-timescale data products in the introduction, such that they appear less unexpectedly in the contributions.

2.3. DATA AND METHODS

2.3.1. p.4 l.92-94: The distinction into training, validation and test is not fully clear to me. You use the validation period to evaluate different architectures and to select model hyperparameters. Could you elaborate on the reason why the evaluation of architecture and the hyperparameter selection cannot/should not be done during the training period?

Using a three-way data split is standard practice in machine learning. This is so that hyperparameters are not chosen based on the test set, which would be cheating.
The purposes of the different periods are as follows:

- The **training period** is used to fit the model, given a set of hyperparameters.
- This model is then applied to the **validation period** to evaluate its accuracy on previously unseen data. The training and validation periods can be used to adjust the model. To find the “best” hyperparameters, this process is repeated a number of times (with different hyperparameters) and the model that achieves the best validation accuracy is selected.
- Only once a final model is selected, we apply it to the **test period**. This way, the model has never seen the test data before and we can be sure that we didn’t overfit our model to the benchmarked metric.

If we had only a training and a test period and selected the hyperparameters on the training period, our model would likely not generalize to unseen data (such as the test period), because we only ever focused on modeling the training period. In the extreme, we would train our model to memorize every data point of the training period and get a perfect fit but terrible accuracy on any other data.

We have extended the paragraph in the revised manuscript to explain the purposes of the training, validation, and test periods in more detail.

2.3.2. p.4 l.101ff: Can you describe the datasets used in NWM and the basic characteristics (e.g. spatial application range, calibration strategy and performance) of the v2 reanalysis product?

Unfortunately, this information is not fully public. We will add the following paragraph to the description of NWM: “For these predictions, NWM was calibrated on meteorological NLDAS forcings for around 1500 basins and then regionalized to around 2.7 million outflow points (source: personal communication with NOAA scientists).”

2.3.3. p.5 Fig1: please also mention what "x" and "+" represent

They represent element-wise multiplication (“x”) and addition (“+”). We will clarify this in the revised manuscript.

2.3.4. p.6 l.127-130: I am particularly interested in how you tuned these parameters and how you decided which parameters to adjust and which ones not. As you mention, the LSTM application is computationally expensive and parameter selection and ranges are therefore important. Therefore, I would rather want to see Appendix D in the main text, and include information why certain parameters are time step dependent and others not.

The decision which parameters to adjust and within which bounds to adjust them is unfortunately largely based on personal experience with LSTMs. To our knowledge, there exist no rules that are universally applicable and agreed upon for the hyperparameter tuning with these models (beyond basic principles like train-validation-test splits, see 2.3.1).

As for timestep-dependent vs. -independent parameters: Are you referring to hyperparameters that are (or can be) different for each *timescale*? In theory, with MTS-LSTM, one could use different parameters for each LSTM branch (e.g., hourly and daily branch) with respect to almost all hyperparameters. The question therefore is for which hyperparameters it may make sense to use different values in the different branches. E.g., for hidden size, we do not see a reason to choose vastly different sizes for the different timescales, since each LSTM branch models a similar process. The only hyperparameter that we *did* choose per timescale is the sequence length, because it defines the point in time where the daily branch hands off to the hourly branch.
For sMTS-LSTM, there are some additional restrictions: Since the daily and hourly LSTM branches use the same weights, they cannot have different hidden sizes (because that would entail different amounts of weights).

Given these considerations, unless there is strong opposition from the editor or reviewers, we would prefer to keep Appendix D in the appendix. Since the concrete hyperparameter choices are very particular to our evaluated models and setup, we think keeping them in the appendix helps to avoid the impression that these may be universally applicable choices for LSTMs (even a dataset of different size may lead to other parameters being better suited). A detailed description of the possibilities, pitfalls, and empirical experience of hyperparameter tuning would be material for a publication in itself.

Also, In Table D1, it seems you ended up with 336 hrs sequence length for both architectures. Would an even longer sequence length lead to better results? What is the tradeoff between higher sequence lengths and computational costs?

Longer sequences do not necessarily lead to better results. The longer the hourly input sequence, the longer the overall input time series will be (because the transition from daily to hourly inputs will happen earlier). Such longer time series are harder for LSTMs to process, because they need to learn dependencies across many time steps. Further, the additional time steps will increase the computational demand of the model. In the extreme, if we'd use an hourly sequence length of 365*24, the hourly branch in the MTS-LSTM would require as much computation as the naive hourly LSTM.

Conversely, shorter hourly input sequence lengths reduce computational complexity (because more of the input is processed at the daily resolution, leading to shorter time series). A too short hourly input sequence, however, will remove information from the inputs that is necessary to generate high-resolution hourly predictions. That said, in our experiments, we did not observe high sensitivity of the model accuracy with regards to the exact hourly input sequence length.

2.3.5. p.6 l. 146-156: Could you explain why these two different LSTM architectures were developed? What are the expected advantages/disadvantages?

sMTS-LSTM is a special case of MTS-LSTM where the different LSTM branches are actually the same LSTM (they have the same structure and weights, see answer to 2.3.6), and therefore they model the same input--output relationships. A-priori, it seems reasonable that some relationships that govern daily prediction hold, to some extent, for hourly predictions as well. Hence, it may be easier to learn these dynamics in a single LSTM branch that processes the different timescales (as done in sMTS-LSTM) than to learn them multiple times, once in each branch (as done in MTS-LSTM). On the other hand, however, there are also differences in how daily vs. hourly data are processed, and these may be easier to learn in a branch that focuses on one timescale (MTS-LSTM) than in a branch that's shared across timescales (sMTS-LSTM).

We agree that the current description does not motivate the two MTS-LSTM variants, and we will include this reasoning in the revised manuscript (formerly L136-156). The following revised text also clarifies the differences between MTS-LSTM and sMTS-LSTM (see questions 2.3.6, 2.3.7):

"The first model, shared multi-timescale LSTM (sMTS-LSTM), is a simple extension of the naive approach. Intuitively, it seems reasonable that the relationships that govern daily predictions hold, to some extent, for hourly predictions as well. Hence, it may be possible to learn these dynamics in a single LSTM that processes the time series twice: Once at a daily resolution and again at an hourly resolution. Since we model a damped system, where the resolution of long-past time steps is less important, we can simplify the second (hourly) pass by reusing the first part of the daily time series. This way, we only need to use hourly inputs for the more recent time steps, which yields shorter time series that are easier to process. From a more technical point of view, we first generate a daily prediction as usual---the LSTM ingests an input sequence of T_D time steps at daily resolution and outputs a prediction at the last time step (i.e., sequence-to-one prediction). Next, we reset the hidden and cell states to their values from time step T_D-T_H/24 and ingest the hourly

input sequence of length T_H to generate 24 hourly predictions that correspond to the last daily prediction. In other words, we reuse the initial daily time steps and use hourly inputs only for the remaining time steps.
In summary, we perform two forward passes through the same LSTM at each prediction step: one that generates a daily prediction and one that generates 24 corresponding hourly predictions. Since the same LSTM processes input data at multiple timescales, it needs a way to identify the current input timescale and distinguish daily from hourly inputs. For this, we add a one-hot timescale encoding to the input sequence. The key insight with this model is that the hourly forward pass starts with LSTM states from the daily forward pass, which act as a summary of long-term information up to that point. In effect, the LSTM has access to a large look-back window but, unlike the naive hourly LSTM, it does not suffer from the performance impact of extremely long input sequences.

The second architecture, illustrated in Fig. 2, is a more general variant of the sMTS-LSTM that is specifically built for multi-timescale predictions, hence, we call it the *multi-timescale LSTM* (MTS-LSTM). Its architecture stems from the idea that the daily and hourly predictions may behave so differently that it is challenging for one LSTM to learn both dynamics, as the sMTS-LSTM would have to. Instead, it may be easier to process the inputs in individual LSTMs per timescale. To reuse daily processing steps towards hourly predictions, MTS-LSTM does not perform two forward passes (as sMTS-LSTM does).Instead, it splits an individual hourly LSTM branch off of the daily LSTM after the initial daily time steps (see Fig. 2). Expressed more technically: we first generate a prediction with an LSTM acting at the coarsest timescale (here: daily) using a full input sequence of length T_D (e.g., 365 days). Next, we reuse the daily hidden and cell states from step T_D-T_H/24 as the initial states for an LSTM at a finer timescale (here: hourly), which generates the corresponding 24 hourly predictions. Since the two LSTM branches may have different hidden sizes, we feed the states through a linear state transfer layer (FC_h, FC_c) before reusing them as initial hourly states. In this setup, each LSTM branch only receives inputs of its respective timescale, hence, we do not need to one-hot encode the timescale. This architecture makes it clear why we call the other variant "shared" MTS-LSTM.

Effectively, the sMTS-LSTM is an ablation of the MTS-LSTM: One can see the sMTS-LSTM as an MTS-LSTM where the different LSTM branches all share the same set of weights and the states are transferred without any additional computation (i.e., the transfer layers are identity functions). Conversely, the MTS-LSTM is a generalization of sMTS-LSTM: Consider an MTS-LSTM that uses the same hidden size in all branches. In theory, this model could learn to use identity matrices as transfer layers and to use equal weights in all LSTM branches. Save for the one-hot encoding, this would make it an sMTS-LSTM."

The last sentence is crucial for the understanding of the differences, I believe "weights of the sMTS-LSTM are shared across all per-timescale branches and its state transfer layers are identity operations." What is an identity operation?

An identity operation is a function that outputs the input value(s). We will rephrase this to “states are transferred without any additional computation (i.e., the transfer layers are identity functions)“ which we hope to be clearer. For more on the difference between MTS-LSTM and sMTS-LSTM, see our answers to questions 2.3.6/2.3.7.

2.3.6. p7. Figure 2: I understood from the text that both the sMTS-LSTM and MTS-LSTM are branching out at each day into hourly predictions. The MTS-LSTM predicts 24 hours, using 72hrs sequence length. Is this the same for the sMTS-LSTM?

It is correct that both MTS-LSTM and sMTS-LSTM predict 24 hours using 72 hours of hourly input sequence length. The first part of the statement (“both the sMTS-LSTM and MTS-LSTM are branching out at each day into hourly predictions”) could be misunderstood: For the prediction of any given day, the hourly LSTM branches off of the daily LSTM only *once* (72h before the last time step). But, if we predict subsequent days to obtain a time series of predictions, the branching point will shift by one day as the predicted day moves forward. All of this holds for both MTS-LSTM and sMTS-LSTM .

The difference between sMTS-LSTM and MTS-LSTM is difficult to understand from just the figure caption. I think it would help to construct the illustration for both architectures to visualize the differences, if possible including the different weights for the MTS-LSTM and the similar weights for the sMTS-LSTM in the diagram.

Unfortunately, we could not find a good way to illustrate the difference between MTS-LSTM and sMTS-LSTM. The basic idea of shared weights is that the daily LSTM branch will behave identical to the hourly branch (if applied to the same inputs). Since the LSTM blocks have a complex internal structure (depicted in Figure 1), it is hard to explicitly show the model weights.

Maybe an alternative perspective on sMTS-LSTM can clarify the setup:

Another way to think of sMTS-LSTM is a single LSTM without any branches. The model works as follows:
1) First, we add timescale flags to the input data:
- We concatenate each timestep of the daily inputs with a one-hot encoding of “daily timescale” (e.g., a vector (1, 0)^T).
- We concatenate each timestep of the hourly inputs with a one-hot encoding of “hourly timescale” (e.g., a vector (0, 1)^T).
2) Then, we ingest the full daily input sequence into the LSTM. This gives us a daily prediction.
3) Next, we re-initialize the LSTM with the hidden and cell state from 3 days ago.

4) Finally, we ingest the last 72 steps of hourly data into the LSTM. This gives us 24 hourly predictions.
Described from this angle, the differences to MTS-LSTM are:
- MTS-LSTM does not need step (1), since there is no need for timescale flags.
- With MTS-LSTM, steps (3) and (4) operate on a *different* LSTM than step (2).

Our rephrased descriptions of MTS-LSTM and sMTS-LSTM (see 2.3.5) should be clearer on these differences.

2.3.7. p.7 l.158: I don't understand why the MTS-LSTM is more flexible in terms of input data than the sMTS-LSTM. In the sMTS-LSTM section you write (p.6 l.139): "we....ingest the hourly input sequence of length TH to generate 24 hourly predictions that correspond to the last daily prediction." Looking at Fig 2, to me this is similar in the MTS-LSTM, where the daily forcings have an effect until the hourly branch starts and then no update using the daily forcings/predictions seems to be made in the hourly branch. Therefore, effectively, you use the daily data until the model branches out and then you use the hourly forcings only? Again, I think it would help to show both architectures in Fig 2.

Our explanation to questions 2.3.5 and 2.3.6 probably clarify this. The reason why sMTS-LSTM cannot use different data products (or even different amounts of data products) for different timescales is that steps (2) and (4) use the *same* LSTM. A single LSTM has a fixed input dimensionality and is therefore unable to process input vectors of varying size. Different meteorological data products may have different numbers of variables, so they cannot be processed by the same LSTM. Since missing explanations from our side seem to inhibit understanding of this point, we include a better description of the differences as outlined in our answers to 2.3.5/2.3.6.

2.3.8. p.8 l.170-184: If I understand it correctly, adding the term into the loss function 'encourages' the model to minimize the difference between daily and sub-daily simulation. But similar to the NSE, this ideal value may not be reached, ending up with a model that is not consistent - even if you put an exceptionally high weight on the mean squared difference?

Yes, this is exactly how the loss works (though the higher you weigh the difference, the more likely will the model learn to generate consistent predictions---but this will come at the cost of poor predictions: e.g., predicting constant 0 is consistent but not useful).

Is there a reason why you don't 'force' consistency across timescales? E.g. when looking at Figure 2 I imagine you could add a function (e.g. simple multiplication of a term) that scales either the daily or the sub-daily prediction (or the average between the two) so that both match the consistency criteria (I now notice that may be similar to what you did in "B1 Delta Prediction")?

Enforcing consistency is in principle possible. And as you note, we tried a number of approaches to achieve this in Appendix B. However, in our experiments, these approaches yielded worse predictions than MTS-LSTM, so we did not further pursue them.

2.3.9. p.9 Table 2: it is a bit confusing to have these different sequence lengths. In the previous section it is 72hrs, here 168hrs, in Table D1 it is 336hrs. Can you harmonize this or explain why there are these differences?

We will clarify this in the revised manuscript.The reasons for the differences are the following:

The 72h in Figure 2 are just for the purpose of illustration: In order to keep the figure tidy, we did not want to show too many “LSTM boxes” after the split into daily and hourly LSTM, so we decided on three days, which translates to 72 hours.

The 336h in Table D1 is what we use in most experiments. This is the outcome of our hyperparameter search for daily--hourly prediction.

The 168h in Table 2 are part of a model that demonstrates how MTS-LSTM can be used beyond daily--hourly modeling and also predict other timescales. The concrete input sequence lengths in Table 2 are somewhat arbitrary (and not hyperparameter-tuned), since our point in this section is less to achieve the best possible NSE but rather to show the flexibility of MTS-LSTM.

We will clarify this by adding “we chose this value for the sake of a tidy illustration; the benchmarked model uses $T^H=336$” to the caption of Fig. 2 and “The specific input sequence lengths are chosen somewhat arbitrarily, as our intention is to demonstrate the general capability rather than to achieve the best possible NSE.” to the caption of Table 2.

2.4. RESULTS

2.4.1. p.9 l.210: that means running ten seeds based on the parameterization in bold in Table D1? If so, I’d add this here.
Correct. We will add this in the revised manuscript.

2.4.2. p.9 l.219: I find this particularly interesting when thinking about hydrological processes. The model parameter values (hidden and cell states) of the last coarse time step (Td - Th/24) are basically your boundary condition/initial state for the hourly model. It seems a bit counterintuitive that the sMTS-LSTM performs better than the naively trained full hourly LSTM. So the ’error’ you introduce through the daily average initial state must be insignificant (due to a sufficiently long sequence length?).

Yes. This is a nice way to think about the modelling system and relates to our motivation of the MTS-LSTM architecture. The idea is that in a damped system, early time steps do not need to be processed at the high target resolution.

Particularly in small basins and for flood peak prediction, this may not always be the case. A plot showing the spatial differences in performance between the naively trained LSTM, the sMTS-LSTM and MTS-LSTM (e.g. similar to Fig 4) could reveal if/where these differences exist. I'd however not be surprised if this plot will show no pattern due to input data uncertainty and randomness in the LSTM and the small performance difference between the LSTM types.

Figure 1 below shows the spatial patterns of the difference between the NSE of sMTS-LSTM and Naive (hourly) predictions together with basin size (indicated by the marker size). As predicted by the reviewer, we cannot see any outstanding patterns that would indicate relationships between basin size and NSE difference.

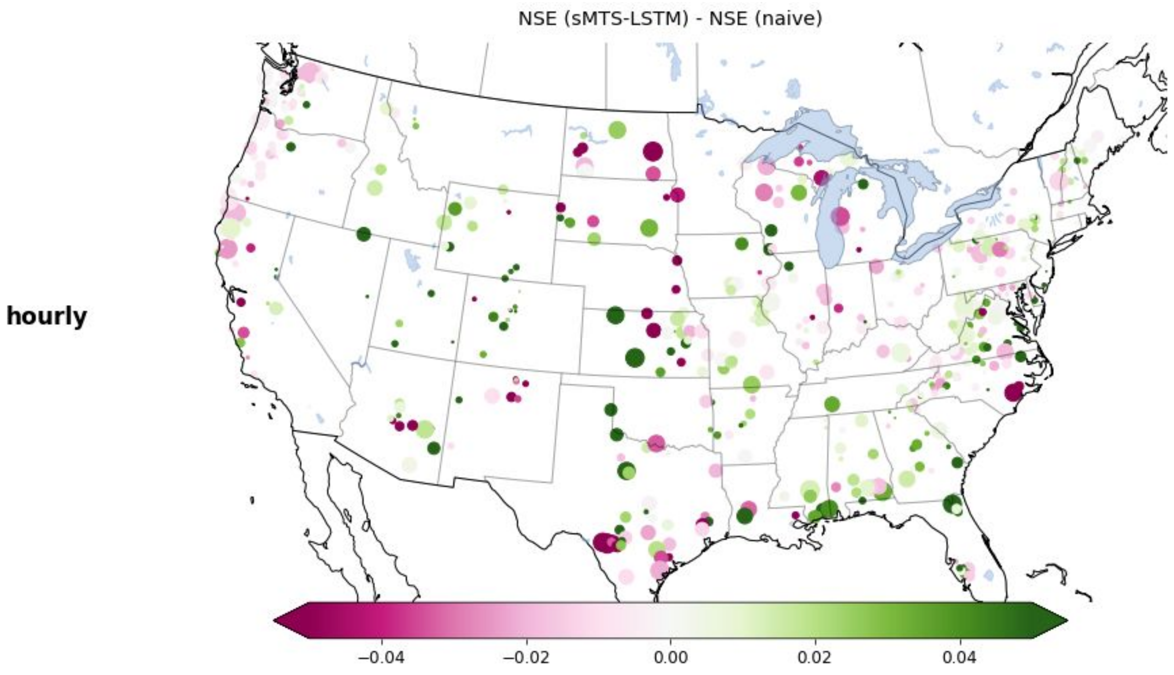

Figure 1: NSE differences between sMTS-LSTM and Naive (hourly). Marker size encodes basin size.

2.4.3. p.14 l.237-250: Interestingly, the Naive LSTM deviates most - probably because the sMTS-LSTM and the MTS-LSTM use recent states from the daily model and are therefore 'closer' to the daily model's flow (volume) prediction?

Yes, this is a plausible assumption, since the shared states will likely make it easier for the model to minimize the consistency term (in some sense one might see the shared states as an inductive bias towards consistency). Unfortunately, we do not see a way to prove it.

The beneficial influence on the NSE could arise because you are introducing a 'physically plausible' constraint in the model which 'helps' adapting the network to the processes? (see also my comment to p8. l.170-184). That is an interesting prospect and if true, could mean adding more of such physical constraints (e.g. global water balance closure) could improve the LSTM even further?

We agree with this line of reasoning and will include it in the revised manuscript. The physically plausible architecture might benefit the predictions. In machine learning, this is sometimes referred to as “inductive bias”. There is indeed ongoing work in hydrology to add such constraints in the form of physical constraints or adaptations into ML/LSTM models (for water balance, we’d like to refer to [1]). However, to our knowledge, so far no modification has improved the overall model performance w.r.t. the NSE.

2.5. CONCLUSIONS

2.5.1. p.16 l.292: it depends on how the NWM was calibrated and what the main purpose is (see also comment to p.4 l.101ff)

We refer to our answer to question 2.3.2 on details of the calibration procedure. Unlike the LSTM-based models, NWM was calibrated only for hourly and not for daily predictions, which may affect the accuracy. This, however, only corroborates our point: Had NWM been explicitly calibrated for daily predictions, we’d expect *equal or better* daily NSEs---and therefore the gap between daily and hourly quality would only grow. Further, since the model was calibrated w.r.t. NSE (like the LSTM-based models) by people who are experts in its usage (NOAA scientists), we would argue that our performance comparison is valid.

2.5.2. p.17 l.293: I understand and agree. But given that LSTMs perform so well for hydrological modelling, efforts should be made to generalize the hyperparameter values for different time steps. I believe you were not sufficiently confident with your tests to deduce general rules for the hyperparameter settings (and that may be a reason why this analysis ended up in the Annexe). But I think it would help the future application of LSTMs if you could give a summary of your experience: e.g. which parameters are time-step dependent, should a parameter increase or decrease with increasing/decreasing time steps, what if someone applies an even coarser time step (monthly)?

First, we would like to reiterate our comment from 2.3.4: We are unaware of any non-trivial universal rules on hyperparameter selection, and would therefore like to avoid the impression that our final parameters are ideal for other tasks and datasets. That said, researchers who work on a similar task and dataset could certainly take our parameters as a starting point for their own tuning procedure.
More specifically to the point of dependence on timescales, only hidden size and sequence length seem meaningful to choose per timescale (though one could maybe come up with scenarios where different learning rates make sense, but we didn’t explore this). As stated in the answer to question 2.3.4, we did not see strong sensitivity with regards to the choice of sequence length (but, again, to achieve the best possible NSE, one will have to

hyperparameter-tune the model to the specifics of the application). For hidden size, we do not see a reason to choose vastly different sizes for the different timescales, since each LSTM branch models a similar process.

2.5.3. p.17 l.296-298: I know the differences are not statistically significant, but can you speculate on why the models are ranked in that order? Somehow the naive hourly LSTM seems not to be able to use this additional information content, or the half year sequence length is not sufficient to depict all states (e.g. groundwater storages may need longer sequence length in some catchments)?

Unfortunately, we cannot make any conclusive statements. One plausible explanation could be that the long input sequences make it harder for the naive LSTM to learn all relationships that exist in the data, while sMTS-LSTM only needs to derive relationships from shorter time series (and the lower resolution doesn't matter much, since it's only low for time steps far in the past). A theoretical explanation could be given by the vanishing gradient phenomen, which is the reason why it is hard for LSTMs to learn dependencies over very long sequences (e.g., more than 1000 time steps).

2.5.4. p.17 l.299-305: Can you speculate why the daily forcings to the hourly MTS-LSTM improve the performance?

We believe that this has essentially the same reason why multiple daily forcings improve daily predictions (already known from previous publications, [2]): Each data product has its individual errors, and given multiple products, the LSTM can intelligently combine the information to counteract these errors.The fact that in this case we use daily data for hourly predictions might reduce this impact, but clearly it does not fully remove it. This may be supported by some degree of smoothness across time: If the daily product says there is low temperature, most (if not all) hours will have had low temperatures, too.

2.5.5. I believe there is more research to be done that you can mention here? E.g. a thorough investigation of time step-dependency of hyperparameters, find measures to use physical constraints in the LSTM (e.g. the regularization)

We agree that there is more research to be done, and we'll add a sentence on physical constraints in the revised paper. Other areas of future research include exploring the potential of uncertainty estimation at multiple frequencies (as opposed to point estimates) and the exploration of architectures that pass information not just from coarse to fine timescales, but also vice versa (similar to our preliminary ResNet experiments that we report in the appendix).

3. TECHNICAL CORRECTIONS

3.1. once introduced, you can stick to the abbreviations (e.g. NWM, MTS-LSTM)

We'll change the revised manuscript to consistently use the abbreviations after their introduction.

3.2. p.3 l.58-60: I think you can refer to Appendix C here

We'll add the reference in the revised submission.

3.3. p.5 l.118: ...half a year...

Yes, we'll change this.

3.4. p.8 l.191-192: it is uncommon to mention results in the methods

Agreed, we'll remove the sentence on results.

3.5. p.8 l.199: this link is supplied here for the third time. Not sure if this is how HESS wants to have references to URLs.

Agreed, the repeated footnote is not necessary. We'll remove it and change the footnotes to citations as per HESS standards.

3.6. p.9 l.215: ’even the naive ones’ - the naive LSTM acts as a benchmark, so it is expected it performs better than (s)MTS?

Yes and no: Yes, it is a benchmark (in the sense of being the most straight-forward way of achieving hourly predictions with LSTMs). But, as explained in the Methods section, we expected the hourly naive LSTM to be problematic since it has to work better or worse than MTS-LSTM.

3.7. p.9 l.216: I think it is fair to add that this worse performance on hourly is much more visible at the NWM

We'll add this in the revised manuscript.

3.8. p.17 l.311-312: I find this first sentence difficult to understand. If possible, split in two

We'll split the sentence and slightly rephrase to clarify.

4. REFERENCES

[1] Hoedt, P.-J., Kratzert, F., Klotz, D., Halmich, C., Holzleitner, M., Nearing, G., Hochreiter, S., Klambauer, G.: MC-LSTM: Mass-conserving LSTM, https://arxiv.org/abs/2101.05186, arXiv, 2021.

[2] Kratzert, F., Klotz, D., Hochreiter, S., and Nearing, G. S.: A note on leveraging synergy in multiple meteorological datasets with deep learning for rainfall-runoff modeling, Hydrol. Earth Syst. Sci. Discuss. [preprint], https://doi.org/10.5194/hess-2020-221, in review, 2020.