EUROPEAN CONSORTIUM FOR MATHEMATICS IN INDUSTRY

ECMI Modelling Week 2023
University of Szeged, Bolyai Institute

Application of data-driven models in hydrological forecasting

Zsolt Vizi (University of Szeged, Hungary)
Péter Kozák (Water Management Directorate of Alsó-Tisza, Hungary)

Auer Lorenz (JKU, Austria)
Endrész Balázs (BME, Hungary)
Mitlasóczki Endre (University of Szeged, Hungary)
Scharnreitner Franz (JKU, Austria)
Tammi Aleksis (University of Eastern Finnland, Finnland)

August 31, 2023

August 29, 2023

Contents

1 Introduction

Szeged is often called City of Sunshine, and yet the people from Szeged often have to worry about too high water levels. The Tisza river flowing through the city, has historically brought several devastating floods to the citizens of Szeged. The goal of our project was to create an algorithm that can predict the water levels for the next week. Of course there have been several, very promising attempts at this before, each with its own benefits and flaws. Some of the tricky aspects of this problem include the ever changing river bed, the very shallow slope of the Tisza and incomplete data due to measurement stations being constructed/removed. In a recent paper[1], co-authored by our instructor, they used an ML model to accurately and very cost efficiently predict the water levels in Szeged for the next 7 days. During the ECMI Modelling Week 2023 our team of five students, with the lead of our instructor Zsolt Vizi, further explored this approach, leading to promising and interesting results.

2 Report

The data, our instructors provided us, consisted of water level measurements at several points of the river Tisza before as well as after Szeged. Figure 1 shows a map of all the 56 measurement stations (blue, green, red) in the original dataset, as well as several upstream dams (yellow). Most of the stations are along the main stem of the river Tisza, which originates in the north of Szeged. However there are also several stations along tributaries. Notably the dataset also features 3 stations past Szeged. These are of importance as the slope of the Tisza is very shallow. If the Danube, which the Tisza flows into, carries more water than usual, the water of the Tisza can not discharge at its usual rate. This is the reason why the floods of the Tisza river often last several weeks. The dataset contains daily measurements starting in 1951 up until 2020. The two biggest floods contained in the dataset were in the years 1970 and 2006, each of them lasting roughly one month.

2.1 Overview

We devided our work into several stages. Each of which will be described by one of the following chapters in more detail.

1. cleaning the data
2. splitting the data
3. data imputation
4. reducing dimensionality
5. choosing a model

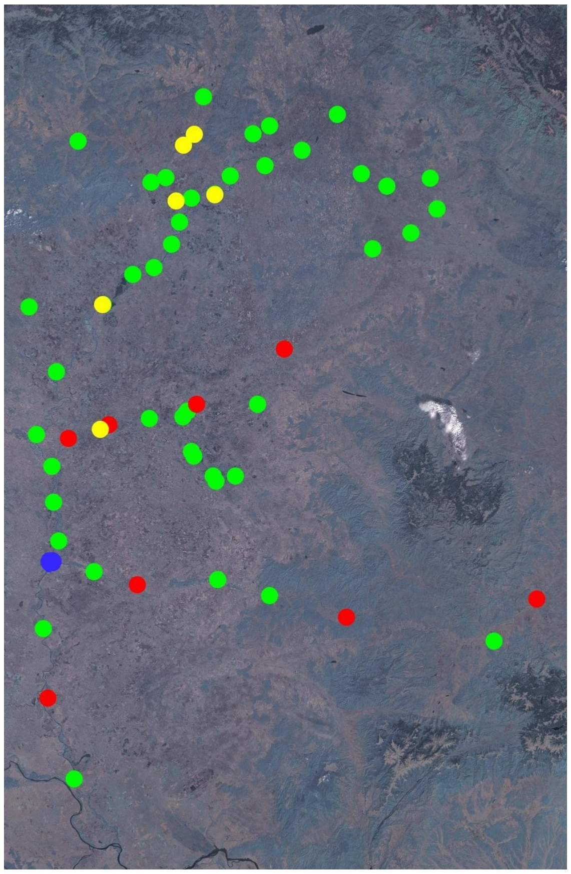

Figure 1: Map of measurement stations (green and red), dams (yellow) and Szeged (blue)

6. implementation

7. verification of results

2.2 Cleaning the data

The dataset, which contains the daily water levels of each station, has some holes. Maybe a station was not yet built, the station was destroyed in a flood or the worker noting the values was inconsistent. Also the river changes over time and old data might not be as reliable as one might hope for.

Due to this we want to choose the stations as well as the time frame of data to use, for training and validation, carefully. We also need to find a way to fill the remaining holes in the data. (see chapter 2.4)

As we want our model to be practical, we have to exclude all measurement

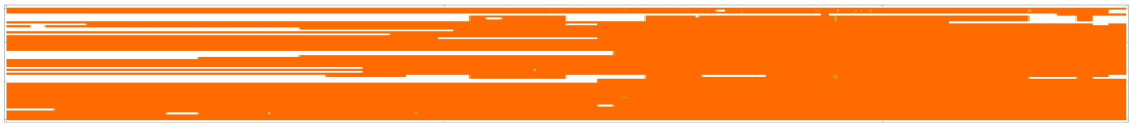

Figure 2: Visual representation of the holes in the data from 1951 (left) to 2020 (right)

stations, which are not in use today. Also if a station started recording no more than several weeks ago, its data will not be useful right now, as we would have to fill in most of its holes. Data, we fill the holes with, will at best not influence the result. There is no way for filled in data to provide new, useful and correct insights.

Following this logic, we exclude stations with very big amounts of missing data. The threshold we choose is more than 50% missing datapoints. This leaves us with a reduced dataset of 48 stations.

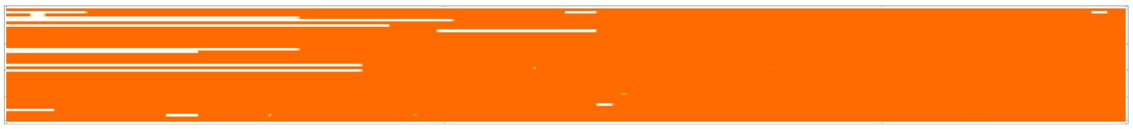

Figure 3: Visual representation of the holes in the reduced dataset

Note that there are still holes in the dataset. If we would cut every station with incomplete data, that would only leave us with around a dozen stations.

2.3 Splitting the data

As the riverbed constantly changes (e.g. erosion, floods, dams), we want to choose the training data with that in mind. For instance, if we would do the usual 80/20 spilt of training and validation data, so, train on the older 80% and validate on the newer 20%, then the model mostly knows how to predict floods from 50 years ago. Would it perform much worse with newer floods, if the river system had changed?

To get an idea on how to spilt the data, we did research on the dams (table 1) and dams (table 2) of the Tisza.

At the end we picked 1951 - 2004 as our training data and 2005-2020 for validation. This way we will see in our validation step, if the model can predict future floods, even when trained on older data. Also, each part of the data set contains a big flood, which is crucial for training and testing as these contain the most extreme data points.

Name	since	lon.	lat.	Wattage
Gibárti vízerőmű	1903	48.317944	21.1635	1000
Felsődobszai vízerőmű	1911	48.263687	21.084688	940
Békésszentandrási duzzasztó	1942	46.891	20.4995	2000
Kesznyéteni vízerőmű	1945	47.99597	21.033205	4400
Tiszalöki vízerőmű	1959	48.025141	21.307876	12900
Kiskörei vízerőmű	1973	47.492961	20.515569	28000

Table 1: General information about the dams

	from	to	height in cm
1970	May	June	961
2006	April	May	1009

Table 2: Biggest floods in Szeged

2.4 Data imputation

In chapter 2.2 we got rid of some of the biggest holes in the dataset. Yet there are some remaining holes, which we will now try to fill, without imprinting further information onto the dataset. For this imputation step, we will also be able to use data from the stations previously discarded.

We decided to fill the missing data of station A with the data of station B via linear regression, if B is the station with the highest correlated measurements to station A. We choose this strategy in order to keep the added information to a minimum, while keeping the algorithm relatively simple.

The pseudocode for our implemented strategy is the following:

Algorithm 1

```
foreach station $A$ with missing data do
    foreach station $B$ with $\text{days}_{\text{missing}}(A) \neq \text{days}_{\text{missing}}(B)$ do
        calculate and store the Pearson correlation of $A$ and $B$
        if the p-value is greater than a threshold, then
            continue
        end if
        store the station $B_A$ with the highest correlation to $A$
    end foreach
end foreach
foreach station $A$ with missing data do
    use linear regression to calculate the missing data for $A$ with $B_A$, store the result as $C_A$
end foreach
foreach station $A$ with missing data do
    fill in $A$ with $C_A$
end foreach
```

What if A and B share missing days? Only the missing data, which is not shared, can be filled in. However running the algorithm multiple times will fill all holes eventually (this only works as the dataset is sufficiently dense). With this dataset 2 iterations were needed, to fill all holes. The implementation in python can be found in the GitHub repository under `notebooks/prepare_data.ipynb`

2.5 Reducing Dimensionality

We now have complete water level data from 48 stations. Reducing the number of stations considered, will help us train our model more efficiently. For example, we would like to only consider stations, that are close enough to influence the water level in Szeged within 7 days. To do this we applied several statistical approaches:

2.5.1 Shifted correlation

The following algorithm was used to estimate how long it takes station X to influence Szeged.

Algorithm 2
Parameter: station X
Results: $t_X \in \mathbb{Z}, p_X \in [0,1]$

```
for $i \in I = \{-5, \dots, 50\}$ do
    calculate correlation $c_i$ of the water level, between station "Szeged" at
    day $d$ and station $X$ at day $d - i$
end for
$t_X = \mathrm{argmax}_{i \in I}\, c_i$
$p_X = p$-value of $c_{t_X}$
```

If we now visualize t_X for every station on a map (figure 5), we can see if the results appear to make sense. Stations further upstream take longer to influence the water levels in Szeged.

Note: The light dot along the Tisza is right at one of the dams.

2.5.2 Other correlation measures

In the procedure described above we initially used the regular Pearson correlation. We then repeated the procedure with Spearman and Distance correlation. As seen in Figure 6 and Figure 7 the results are very similar.

2.5.3 Correlation $\neq$ Causality

In general, correlation does not imply causality. However, since we are looking at a river we know there exists at least some level of causality since water flows downstream. The results we measured via correlation, also support this direction of informational flow. We even found, that stations downstream of Szeged show negative delay, further supporting the validity of the test performed above.

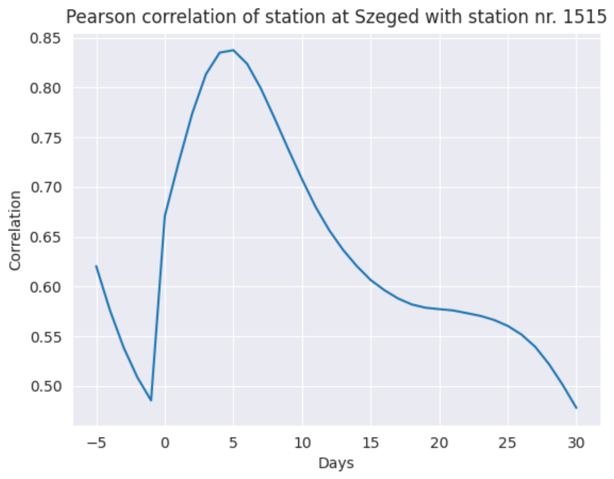

Figure 4: Graph: x-axis: i, y-axis: c_i for station 1515 (see Appendix), $p_{\text{val}}(t_X) \ll 10^{-3}$, 2004-2006

2.5.4 Conclusion

With each correlation measure we got similar results. Except for less than a handful of stations, their maximally correlated delay was 7 days or less. So dropping stations, with the suggested argument of them being too far away, did not work.

2.6 Choosing Models

We mainly considered 3 different models.

i. The **LSTM** (= Long-Short Term Memory) takes the water level data of a single day, as well as its memory, as an input and outputs its new memory for the next day. This is run recursively multiple times. On the last day the network makes a prediction for the next seven days.

ii. One could use a **CNN** (= Convolutional Neural Network) which convolutes over the time dimension of the input. This way the CNN might be able to "see" flood waves coming. Whereas the LSTM has to mentally keep track of them.

iii. Seven **fully dense NNs** that each take only a single measurement per station as an input and each provide a prediction for a single day in the forecast.

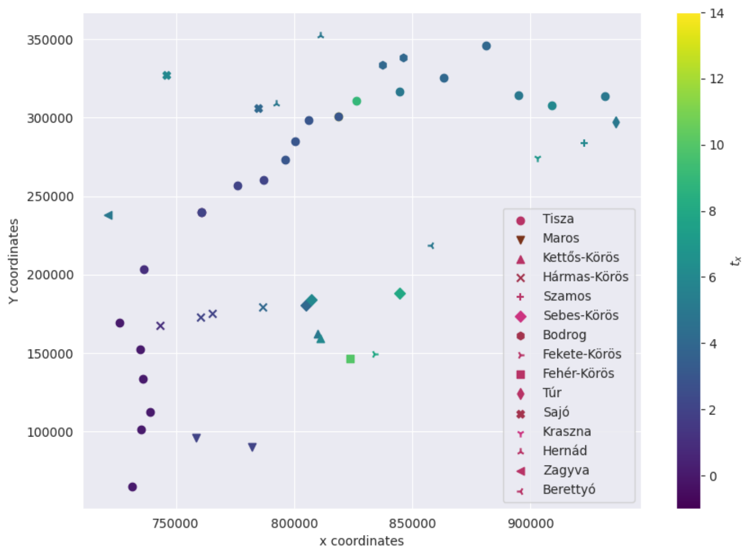

Figure 5: Stations colorized by t_X

2.7 Implementation

All the models were implemented in a PyTorch environment. In the process of making the prediction, using a neural network, a critical step is the data modification and split. In the case of all the models, fifteen days of data from the past was used to predict the following seven days. So the data should be fed to the networks as an input of a $15 \times K$-sized matrix (where K is the number of stations used) and the ground truth data of the following predicted days are a seven-number long vector representing the water levels in Szeged.

In the topic of neural networks, one genuinely significant aspect of learning, is the manipulation of the data. We already applied the steps described in chapters 2.2 to 2.4. Furthermore the data should be standardized, having only a smaller region of input and output data. In the most cases, the standardization means using zero means and unit standard deviation by subtracting the mean of the training data from all data and dividing by the standard deviation of the training data. Which needs to be done for each station separately. It is highly important to use only the training set for determining the mean and standard deviation values, in other cases some validation information can be added to the network, which can cause look-ahead bias.

After the data preprocessing is done, the models can be built.

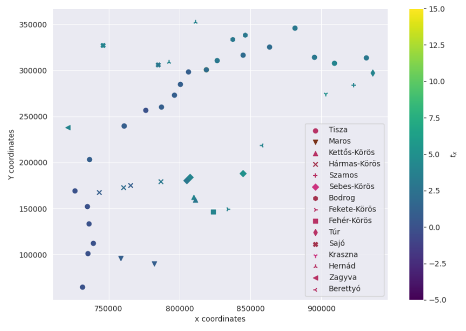

Figure 6: Stations colorized by t_X

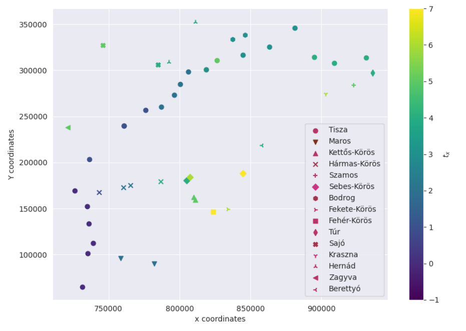

Figure 7: Stations colorized by t_X

2.7.1 LSTM

The LSTM model consists of an LSTM layer, followed by a linear layer at the end. The hyperparameters of the model are ten hidden sizes for the five LSTM layers. The linear one is used only for subtracting the result from these layers, using ten input and seven output neurons.

2.7.2 CNN

The first CNN approach uses only an arbitrary order of stations and an one-dimensional convolution along the dataset, so the convolution kernel is only shifted along the time dimension and uses the data from all chosen stations, added as different channels to the model. There are five-, nine- and three-sized kernels being used respectively, by the seven hidden channels, each separated by a ReLU activation function.

In the case of this model, the network has to understand the spatial information inside the data, because all the stations, in an arbitrary order, get passed to the network, so there is no information added to the network in connection with how fast the water level difference is propagated to Szeged. So, the CNN should use the spatial information as well. To do that, the new dataset is arranged using the causality information in a way, that on the left Szeged and stations with maximal zero-day difference is indicated and going towards the right side of the table, more days take the water affecting Szeged station. This is fed to the network in a form, that in the first convolution layer only the closest stations are added to the network as channels, **after day the stations being one day far away Szeged,** followed by two day one... using seven blocks at the end. The last block contains not only the stations which are eight days away, but all the remaining ones as well. This way the network gets new information at every step whilst processing the old information as well. In the end, the above-described CNN is ended to process all the information.

2.7.3 Dense NN

A dense neural network was also built using one-day data from all the stations. The used day was determined by the spatial information, so always the most significant day was added to the network. In order to keep the task simple, for a fully dense linear network, only one-day prediction was used, and seven different networks were built to predict all seven days. This network is not further mentioned because the results were much worse than the baseline model.

2.7.4 Custom loss functions

Another highly significant part of building a neural network is deciding what kind of loss function is used. We tried several kinds of functions. At first, the widely used L2 loss function, which is the mean square error between the ground truth and the prediction. The test showed that using this many stations will give inaccurate results for the first few days but relatively good ones for the end. To make the network focus better the following days ahead, the first idea was to use a weighted mean square error, having linear or squared weights, by dividing the difference by the number of days being ahead of the current one or by the square of them.

The network should perform well, not only in everyday cases but also when floods occur. This can be done by giving a bigger punishment to the network for not being precise when the water level is higher. The custom loss functions are the following:

- ReLU function giving a smaller weight in the case of the difference is under a specific water level and after that a linearly increasing weight.
- ReLU function with the same property described above but using not a linear but a squared increase.
- Using a sigmoid-like function, built from two different parabolas.

- Staircase function, giving increasing weight as the water level grows.
- Two squared ReLU functions, one for the positive and one for the negative x-axis. This is the variant we use when talking about a custom loss function in the following chapters.

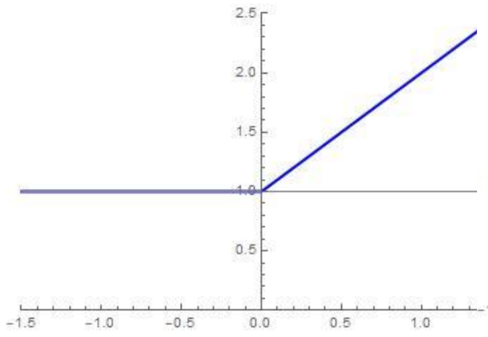

Figure 8: custom loss, ReLU

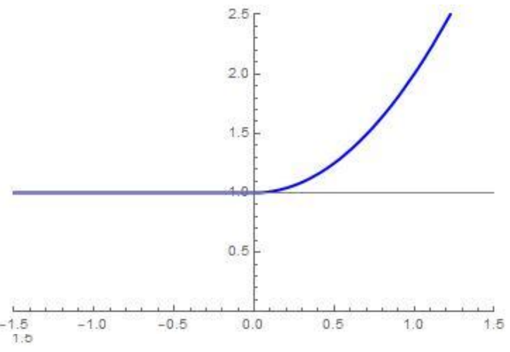

Figure 9: custom loss, squared ReLU

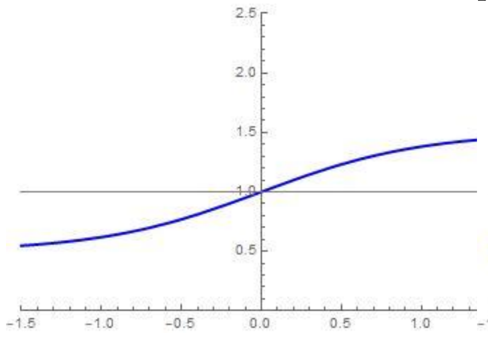

Figure 10: custom loss, sigmoid

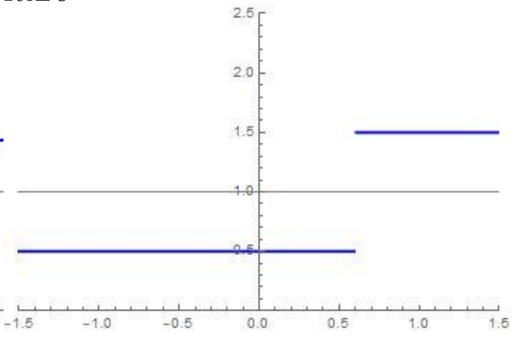

Figure 11: custom loss, staircase

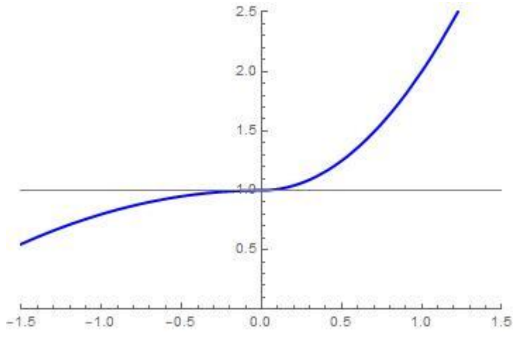

Figure 12: custom loss, double squared ReLU

2.7.5 Learning rate

To have the best results from the networks possible, a learning rate scheduler, which reduces the learning rate in the case of reaching a plateau, was added to the network. We used Adam optimizers and early stopping with ten epochs of patience. All the models were trained in five hundred epochs (but they stopped before then, because of the early stopping feature). The used batch size was eight.

2.8 Verification of Results

We utilized four distinct metrics for the verification of our models. The code employed for this purpose is largely adapted from the existing verification code provided by our instructor. The validation dataset, spanning from 2005 to 2020, was employed for this analysis.

The metrics encompassed the Mean Absolute Error (MAE), offering insights into error magnitudes, and the Root Mean Square Error (RMSE), which emphasizes outliers more than MAE. Additionally, we computed R^2 values and employed Willmott's Index (WI). For the MAE and RMSE values near 0 indicate a better fit. For the R^2 and the WI values close to 1 are preferable.

Our analysis extended to evaluating deviations from the required confidence interval of 5cm over a 5-day period for each model. The results were visualized through scatterplots and tables. Moreover, we examined how the models performed during flood conditions, specifically focusing on the 2006 and 2013 floods. This involved generating line plots that compared model predictions against observed values for the given time period. As a further comparison we also checked a timespan with medium to low water leves, during June and July of 2007.

3 Discussion

The above-described models were compared to each other and two other models, baseline and the best performing LSTM model from the given paper[1]. The results were compared with the same metrics described in the paper. Using MAE, RMSE, R2 and WI metrics, the same result can be seen. For the first-day prediction, our models underperform the baseline model. However, our LSTM and simple CNN give a better result after the fourth day of prediction. An interesting behavior can be seen in the case of the spatial CNN because it gave worse results than the baseline model in the case of the first and last two days. But having an overall view is not enough in our case. A good model can predict well not only on average but in the case of a flood as well.

Using the flood from the validation data we can see that the LSTM network cannot reach the top of the water level in a case of prediction in the first three

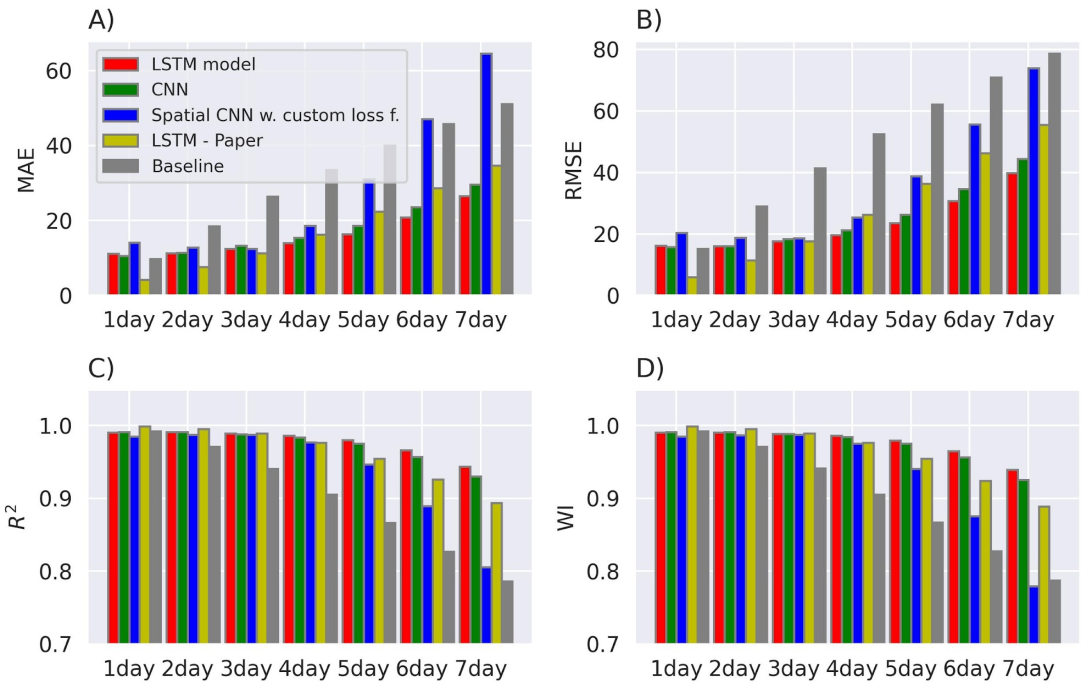

Figure 13: Results, different metrics

days, but it barely reaches the given required interval using one sigma confidence interval for the fifth and seventh day. Though, on average the LSTM network outperformed the simple CNN one, it managed to reach the given required region in the case of a flood. Using spatial information, the same well-performing result can be seen with a smaller confidence interval. The CNN interval shrank the deviation of the data.

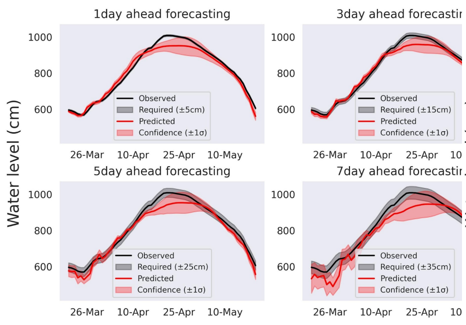

Figure 14: 2006 flood prediction, LSTM

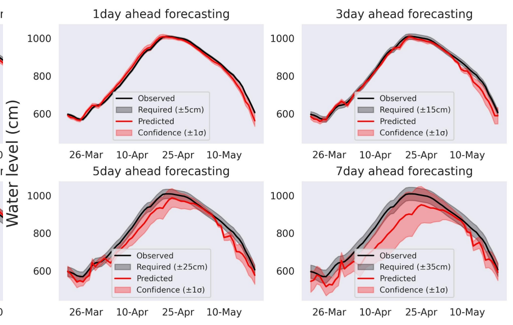

Figure 15: 2006 flood prediction, CNN

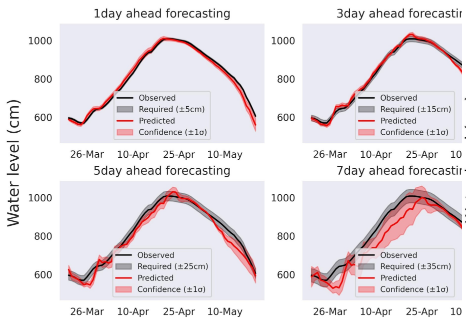

Figure 16: 2006 flood prediction, custom loss CNN

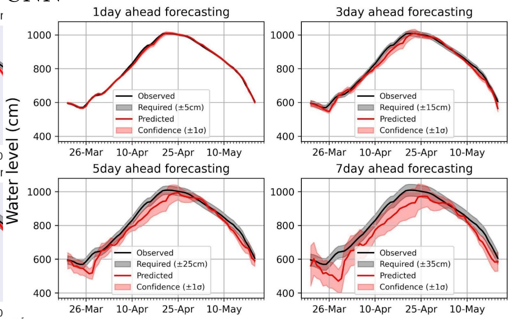

Figure 17: 2006 flood prediction, paper[1]

4 Conclusion

The main difference between our method and the one described in the paper[1], is, that we used information, collected by many more stations, for predicting the model but smaller and simpler models as well. As we can see in the results, using more stations helps in the prediction for the future. We managed to outperform the paper's network after day four, but the prediction was poor, for the first day, even worse than a baseline model.

5 Group work dynamics

During the modelling week, our group participated in meaningful and productive communication. Due to the evolving nature of the project, it was not possible to plan far ahead, and the work was distributed afresh every day. First thing in the morning, we started with brainstorming new ideas and distributing the work. It quickly crystallized itself which tasks suited everyone best. While

Balázs Endrész and Lorenz Auer mostly worked on the machine learning models, Aleksis Tammi and Endre Mitlasóczki worked on both data visualization and the models. Franz Scharnreitner was working on data analysis and imputation. Thus, the work was evenly split, and the group was able to work without much downtime.

During brainstorming, the group had a multitude of ideas, and it was a challenge to filter out the ones that were worth pursuing and manageable within the timeframe of a week. The group handled this challenge very well, although there were still many ideas left behind, that could have been implemented if there was more time to work together.

Sharing code was achieved both through GitHub and Google Colab. Both tools worked well, and we did not have any problems with files not being shared or version conflicts.

6 Instructor's assessment

The work with the team was arranged as an R&D project is structured in the industry. In the morning of the first day, we had a detailed discussion about the topic of the problem: scientific background, data collection methods, structure of the data, previous results and goals for the week. They showed great attention and raised good questions already in the very first discussion, which gave me a good impression about the participants. We planned with three further catch-up sessions during the week (Monday, Wednesday and Friday afternoons), but we had short discussions every day, since they had several, equally promising and creative ideas about data processing and model building, thus I was eager to have more sessions with them.

None of them were an expert in machine learning algorithms applied for time series (which is a complex topic), but they were able to ramp up the required knowledge in a short period of time: this skill will be very valuable at their work in the industry. I was also satisfied with their ability for system understanding, which led to have good and creative improvement ideas at different steps of the machine learning pipeline (data imputation, ML model architecture, loss functions) even in the early stage of the work. Despite there were some flawed or not well thought ideas during the week, we could discuss the problematic points all the time and they were improved them until the next meeting.

Since this project was really implementation-heavy, fortunately everybody was familiar with programming in the necessary level and there were leaders in the coding of the larger blocks of the code. They got used to programming in a collaborative manner in short time, which spared a lot of time for them and this enabled them to try a lot of ideas and execute several model trainings. I was satisfied to see that they can use and build into their code the previously developed research code without bigger issues.

It was clear from the second day, that they were able to discuss the strengths of the individual members of the group and they could distribute very quickly the work of the complex task, which indicated their capability for teamwork. As I have seen there was no real conflict in the team: if someone was responsible for a concept or code, he explained the details, but it was clear that everybody played a part in that specific part. They could also prove their excellent teamwork at the presentation of the Modelling Week, where everybody contributed to the talk.

In summary, I was very satisfied with their work they did during the intensive 5 days of the ECMI Modelling Week. They implemented good improvement ideas for water level prediction and they were able to outreach the previous performance results for later days of the prediction. In my opinion, they got a complete picture about data-driven algorithm development in a team, which is a fruitful experience for their later career in this field.

References

[1] Zsolt Vizi, Bálint Batki, Luca Rátki, Szabolcs Szalánczi, István Fehérváry, Péter Kozák, Tímea Kiss. Water level prediction with various machine learning algorithms for a lowland river. 2023.

A Code

- GitHub repository: data imputation and validation
- Google Colab: work notebook and LSTM
- Google Colab: CNN

B Stations

A table of stations and their metadata can be found in the GitHub repository under `docs/stations.csv`