UTRECHT UNIVERSITY

FACULTY OF SCIENCE

Applied Data Science

Final Thesis Project

Using artificial neural networks to improve hydrological streamflow predictions from PCR-GLOBWB

Oriol Pomarol Moyà

Advisors

Edwin Sutanudjaja (UU), Derek Karssenberg (UU), Youchen Shen (UU)

July, 2022

Contents

Abstract **3**

1 Introduction **3**

2 Data **5**

3 Methods **7**

3.1 PCR-GLOBWB 8

3.2 Error-correction 9

3.2.1 MLR 9

3.2.2 FCNN 9

3.2.3 TCNN 10

3.3 Model setup and evaluation 11

4 Results **14**

4.1 Model variability 16

4.2 Performance consistency 19

5 Conclusion and Discussion **22**

A Appendix **27**

B Appendix **30**

Abstract

This project aimed to improve the streamflow discharge simulation performance of the PCR-GLOBWB hydrology model in the Rhine basin by modelling its residuals using artificial neural networks. Two architectures were used, a more generic fully connected network and a temporal convolutional network, as well as a multiple linear regression as a baseline. The predictors included a bunch of PCR-GLOBWB output variables (e.g. runoff components, groundwater recharge, snow, groundwater stores, etc.) and meteorological input variables (precipitation, temperature and reference potential evaporation), which were fed to the models either directly or by adding lagged versions of them of up to 60 days. The results showed increased performances to the original PCR-GLOBWB simulations, but no significant differences were found between the different machine learning models.

1 Introduction

Streamflow is a most important hydrological variable that can affect agriculture, ecosystems and, ultimately, society. Being able to accurately predict long-term streamflow can lead to an improvement in water resource management as well as flood mitigation.

The use of machine learning (ML) models in hydrology has been rapidly spreading in the past few years due to their high performance and flexibility. C. Shen et al. (2021) showcases many different ML applications in a wide range of hydrology-related topics. Such an increase in popularity is not only due to the increase of computational power but also to the large quantities of data that have become available, which contain significantly more information than hydrologists have been able to capture within theories, according to Nearing et al. (2021). This article also states that, when ML models have been benchmarked against calibrated conceptual models or process-based models, they have generally performed better. Conversely, in many cases when predictions involve scenario analysis (e.g. when evaluating effects of climate change or land-use change on river flow) ML may not function properly. To try and get the best of both worlds, some researchers have implemented a combination of the two approaches within a single modular model, referred to as *hybrid modelling* in the literature (Lange & Sippel, 2020), that can obtain improved forecasts.

Isik et al. (2013), for example, used two fully connected artificial neural networks to predict baseflow and stormflow from the surface runoff as determined by the SCS-CN model as well as other meteorological variables. A much simpler approach was followed by Noori and Kalin (2016), who created a fully connected artificial neural network to improve the streamflow prediction from the baseflow and stormflow outputted by the physically-based SWAT model. Y. Shen et al. (2022) used a Random Forests approach taking an extensive set of hydrological state variables simulated with the PCR-GLOBWB model along with precipitation, temperature and evapotranspiration in a coupled model that corrected the initial PCR-GLOBWB prediction, improving its performance significantly at three streamflow gauging stations in the Rhine basin. This strategy will be quoted as an error-correction model from now on.

In parallel, the adoption of ANNs in hydrology has had a long history (Lange & Sippel, 2020) due to their ability to learn non-linear relationships between variables and find relationships between those and the output without prior knowledge of the physical characteristics of the problem. Maheswaran and Khosa (2012) found that the studied streamflow series from two rivers in India showed significant results for the presence of non-linear features. In the study from Duan et al. (2020), the linear model performed the worst among other neural network alternatives when predicting streamflow from meteorological variables in almost all of the studied basins, which they attributed to the non-linearity of the prediction problem. On top of that, in the same study, the models that took into account the temporal dependency of the data performed even better than a fully connected network, especially the Temporal Convolutional Neural Network. Other articles such as Gao et al. (2010) have also used artificial neural networks to predict streamflow from meteorological data in order to study the impact of different climate change scenarios in hydrology.

This project aims to improve the streamflow predictive performance of the PCR-GLOBWB model by using an ANN-based error-correction model, taking its simulated variables and meteorological data as input in a similar setting as Y. Shen et al. (2022). In that paper, the use of a Random Forests model produced a significant increase in performance from the base PCR-GLOBWB model, which raises the question of whether similar or better results can be achieved with already proven ANN methods.

Two ANN architectures have been employed in this project, a fully connected neural network (FCNN) and a Temporal Convolutional Neural Network (TCNN). While the first is a more generic and widely used ANN architecture, the second takes into account the temporal information from the data and has been shown by Duan et al. (2020) to outperform other common time-series approaches such as LSTM or GRU. To check whether such complex models are worth using, a Multiple Linear Regression (MLR) is also executed as a baseline. The data is fed to the models in two formats; adding their lagged versions up to a certain threshold, or not.

The main research question is then the following: Can artificial neural networks significantly improve the performance from PCR-GLOBWB in an error-correction approach for streamflow prediction in the Rhine basin? Furthermore, four sub-questions grouped into two main topics will be tackled.

1. Are non-linear ANN models significantly better than multiple linear regression for error-correction modelling in streamflow prediction? Are the predictions of such models performing equally well for all streamflow values?

2. Does the introduction of lagged variable information significantly improve the performance of the error-correction models? Can a time-series specific model (TCNN) perform significantly better than other more generic approaches (MLR, ANN)?

In the following sections we will explore the data used in this project, the methods put into practice to respond to these questions, the obtained results and, to conclude, a brief discussion of their implications.

2 Data

The data from this study belongs to two different locations in the Rhine basin: Basel and Lobith. It was obtained from GitHub (Y. Shen, 2021) and contains daily observations consisting of 3 meteorological features and 18 simulated state variables from the PCR-GLOBWB model, for a period of twenty full years, from 1981 to 2000. The data was already clean and complete.

The meteorological variables were precipitation, temperature and reference potential evapotranspiration, all of which have been commonly adopted for both direct streamflow prediction (Duan et al., 2020; Gao et al., 2010) and hybrid models (Isik et al., 2013; Y. Shen et al., 2022).

As mentioned before, the rest of the predictors fed into the machine learning model were the variables simulated by PCR-GLOBWB, which are listed in Table 1. More information on the PCR-GLOBWB model can be found in Section 3.1.

The predicted variable fed to the ML models was the PCR-GLOBWB residual, obtained by subtracting its streamflow prediction from the actual observations of the gauging stations in each location. As can be seen in Figure 1, despite the noise, there is a clear seasonal variation of the observed streamflow that differs between locations. The figure shows only half of the full extent of the data, to allow for easier visualization of the seasonal variability.

In Basel, there is a nival regime characterized by high discharge in summer due to the melting ice, while in Lobith there is a pluvial regime which typically corresponds to higher discharge in winter and spring as a direct consequence of precipitation. The residuals obtained from the uncalibrated PCR-GLOBWB model also show a certain seasonal variation, generally increasing for higher runoff values and thus following an analogous shape to the streamflow observations, especially in Lobith. In Basel, some peaks are inverted in the residuals instead. This seems to imply a certain correlation between the streamflow observations and the residuals or, at least, an increased difficulty from PCR-GLOBWB to correctly predict the high streamflow periods.

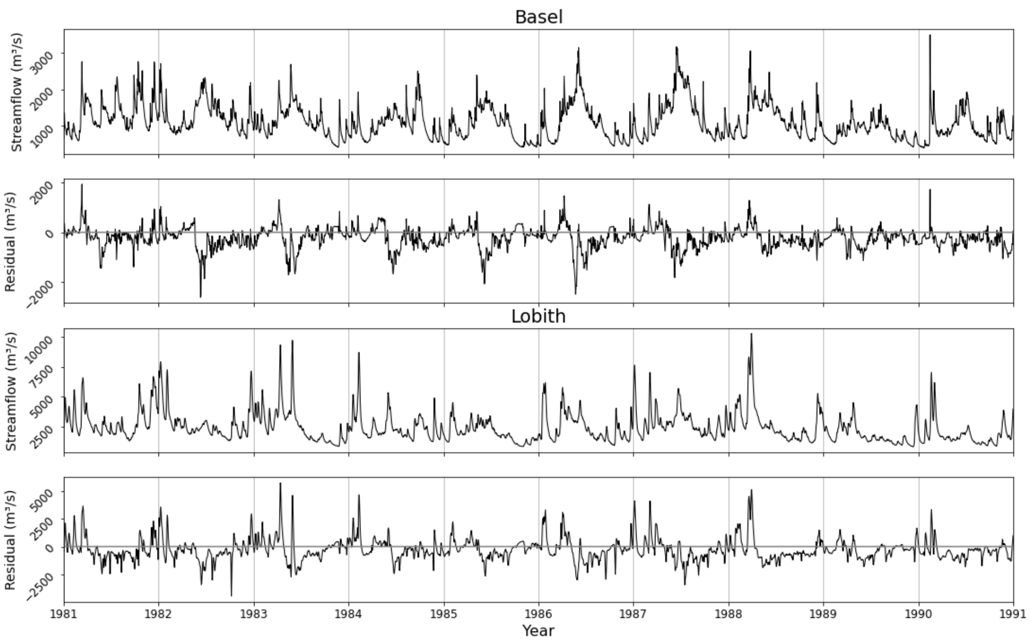

Figure 1: Time series of the observed streamflow and residuals from the PCR-GLOBWB prediction for Basel and Lobith, between 1981 and 1990.

3 Methods

The error-correction approach mentioned in previous sections has been applied in this project in the following way. First, the PCR-GLOBWB model was run for the selected time period and location to obtain its simulated state variables and streamflow prediction as part of the input data. Then, the available data was split between training and testing. For the training set, an ML model was fitted using both the state variables and some additional meteorological variables as predictors, and the residual from the PCR-GLOBWB streamflow prediction as a dependent variable. To make models aware of seasonality, the day of the year was also included as a predictor. For the test set, the output of the ML model was summed to the streamflow prediction from PCR-GLOBWB to obtain the definitive streamflow forecast to be compared in the results.

In the next sub-sections both PCR-GLOBWB and ML models (MLR, FCNN and TCNN), as well as their choice of parameters, are explained.

3.1 PCR-GLOBWB

The PCR-GLOBWB model (Sutanudjaja, 2017) is a grid-based global hydrology and water resources model that can predict streamflow as well as other state variables described in Table 1.

The model consists of three layers, two soil moisture storages and a groundwater storage. Each of those has its own water flow (surface runoff, stormflow and baseflow), as well as water exchange between them, with the atmosphere through the top layer, and with human interaction through industry, livestock, domestic use and irrigation.

Table 1: Simulated state variables obtained from the PCR-GLOBWB model as described in Y. Shen et al. (2022).

Variable name	Unit	Explanation
baseflow	m/day	baseflow, groundwater discharge
directRunoff	m/day	surface runoff
domesticWaterWithdrawal	m/day	domestic water withdrawal
gwRecharge	m/day	groundwater recharge, fluxes from the lower soil layer to groundwater stores
industryWaterWithdrawal	m/day	industrial water withdrawal
interflowTotal	m/day	interflow, shallow sub-surface flow
irrigationWaterWithdrawal	m/day	water withdrawal allocated for irrigation purposes
livestockWaterWithdrawal	m/day	water withdrawal allocated for livestock demand
nonIrrWaterConsumption	m/day	non-irrigation sectoral (domestic, industry and livestock) water consumption, i.e. non-irrigation sectoral withdrawal minus return flow
snowCoverSWE	m	snow cover/storage in water equivalent thickness (excluding liquid part)
snowFreeWater	m	liquid water/meltwater storage in the snowpack
storGroundwater	m	groundwater storage (renewable part)
storUppTotal	m	S1 actual upper soil water storage
storLowTotal	m	S2 actual lower soil water storage
surfaceWaterStorage	m	surface water storage (lakes, reservoirs, rivers, and inundated water)
totLandSurfaceActuaET	m/day	total evaporation and transpiration from land part
storGroundwater	m/day	total evaporation and transpiration from land and water body parts

The model was run with a daily time step and a spatial resolution of 30 arcmins. For the rest, the default parameter values were used since Y. Shen et al. (2022) found that, after applying the error-correction model in each location, the performances of calibrated and uncalibrated PCR-GLOBWB runs were equally good, so this extra step was not necessary.

3.2 Error-correction

As introduced in Section 1, to compare the importance of time-series in streamflow prediction, two ANN architectures were tested; a fully connected neural network (FCNN) and a temporal convolutional neural network (TCNN). Additionally, a Multiple Linear Regression (MLR) was used as a benchmark. All of these models are explained in the following subsections.

3.2.1 MLR

The multiple linear regression is the simplest model of them all since it takes into account only linear dependence between the dependant and each of the independent variables. It was implemented using the *LinearRegression()* method from Scikit-learn (Pedregosa et al., 2011) library using the default setup. Linear regression was also used as a baseline for other ML models in (Duan et al., 2020).

3.2.2 FCNN

A fully connected neural network is a type of ANN, a family of ML models that uses layers of units called neurons. Every neuron takes as input the response from the previous units to which it is connected, computes a weighted sum, and applies a certain activation function to obtain its output. In regression problems, the last layer consists of only one neuron with a linear activation function, the output of which corresponds to the prediction of the model. The weights of the connections between neurons are trainable parameters, optimized for the task at hand through a process called backpropagation. In FCNN in particular, every neuron takes its input from all the neurons in the previous layer. This is a generic structure that can adapt to many scenarios and has been found to improve streamflow prediction when coupled with physically-based models (Noori and Kalin, 2016).

The model was built using Keras (Chollet et al., 2015) and the hyperparameters of the network were optimized using the Hyperband tuner from KerasTuner (O'Malley et al., 2019), using a randomized 20% of the training data as validation and mean squared error (MSE) as the loss metric. For the activation function, after a few test runs allowing the tuner to choose between the two most common options, *relu* and *sigmoid*, it was observed that the former was almost always preferred over the latter. Since *relu* was also the choice in Duan et al. (2020), it was fixed for future runs. The number of hidden layers was also allowed to vary at first between 1 and 3, but it was finally set to 2 for similar reasons. The learning rate was mirrored to that of Duan et al. (2020), fixing it at 0.0005 and using the Adam optimizer. Having all of these parameters remain constant allowed the model to run much faster, a necessity considering the little resources available for this project. The only parameter that was allowed to vary between runs was the number of neurons in each layer, between 25 and 200 in steps of 25. Finally, a 0.05 dropout layer was added after every hidden layer to improve generalizability.

3.2.3 TCNN

Convolutional neural networks are a class of ANNs in which every neuron in any given layer is only connected to a distinct subset of neurons from the previous layer, defined by a sliding kernel.

The temporal convolutional neural networks use this same structure but adapt it to time-series data. The main difference is that, due to the temporal nature of the data, it must use casual convolutions, meaning that a neuron can only receive information coming from past time steps. In Figure 2, this implies that the output neurons can not be connected, directly or indirectly, to any unit to their right. Additionally, it uses dilated convolutions, which reduces the number of connections as depicted in Figure 2, speeding up its training and reducing the model's complexity. For every value of streamflow, a certain time window from every predictor variable must be provided, so the model can learn the temporal patterns of the data. Duan et al., 2020 found that TCNN performed better than other deep network architectures such as FCNN, LSTM or GRU when predicting streamflow, and complimented its potential to perform future hydrology projections.

The Keras TCN (Rémy, 2022) library was used to implement a TCN layer with a kernel size of 3 and dilation of (1,2,4,8). Using these parameters, the resulting receptive field had a size of 61, meaning how far can each output neuron receive information from the input. This value should be bigger than

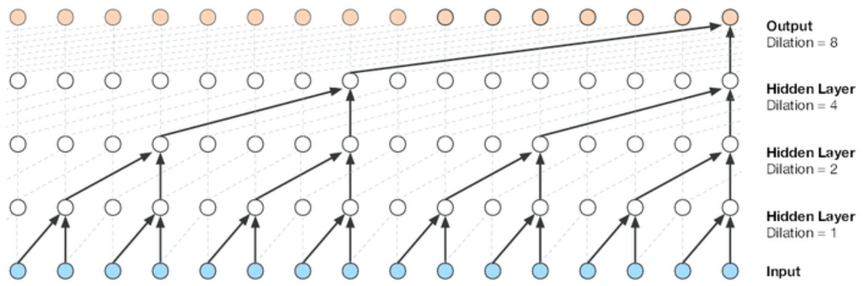

Figure 2: Visualization of a stack of dilated causal convolutional layers (Oord et al., 2016), with a kernel size of 2 and dilation of (1,2,4,8).

the time window so that all of the information provided can be "seen" by the model. Additionally, a dropout rate of 0.05 (as recommended by the author of the package) and layer normalization were applied. The *nb_filters* parameter, which determines the complexity of the model, was optimized for each run with KerasTuner between 25 and 200 in steps of 25.

3.3 Model setup and evaluation

For the input variables, two different settings were tested, including the addition or not of lagged variables (Table 2). In the *no_lag* models, all the variables described in Section 2 were fed directly to the MLR and FCNN models. The *lag* models, on the other hand, were inputted additional lagged versions of all the previous variables to introduce the time-series information to them.

The number of time steps added for the lagged version of the data was of 60 days, on account of a parallel thesis project (Afshari Hemmatalikeykha, 2022) based on the same data, which found that there was still a significant correlation between the past values from some meteorological variables up to this number and the streamflow observations. In this project, it was assumed that the same was true for the residuals, considering they showed some degree of correlation with the streamflow (see Section 2). On the other hand, increasing the lag or fine-tuning it would have increased the running time of the algorithms to unattainable levels given the time and resources available for this project. The lagged version of the data was obtained by adding, for every existing variable v, as many new variables v_{lag_i} as determined by the lag parameter l. These new variables contained the past i_{th} time step for every observation j, as demonstrated in Equation 1.

$$v_{lag\_i}(j) = v(j - i) \qquad i = 1, 2, \ldots, l \tag{1}$$

For TCNN the data must be formatted to follow a different structure that uses a time window instead, but the information available to the model is the same. The first few rows, where no information on the previous time steps is available, had to be discarded, slightly reducing the total amount of observations available for the *lag* models. Each variable was standardized based on training data only, not to overestimate the performance, and the lagged versions of the variables were standardized using the parameters from their non-lagged counterparts.

The metric chosen to assess model performance is the Kling-Gupta efficiency (KGE), which is gaining dominance in recent hydrology literature and is preferable to other popular metrics such as Nash-Sutcliffe Efficiency for streamflow prediction since it can better capture the data seasonality (Y. Shen et al., 2022). To achieve that, it combines three components (Gupta et al., 2009). First, there is the linear correlation between predictions and observations r, second, a measure of relative variability α, and finally a bias term β. From these, the KGE can be obtained as shown in Equation 2. In practice, the KGE was computed using the hydroeval (Hallouin, 2021) library.

$$KGE = 1 - \sqrt{(r-1)^2 + (\alpha - 1)^2 + (\beta - 1)^2} \tag{2}$$

where $\alpha = \sigma_s/\sigma_o$ and $\beta = \mu_s/\mu_o$, and μ and σ refer to the mean and standard deviation from the simulated x_s and observed x_o values.

To evaluate the models, five-fold cross-validation was employed, taking 80% of the data each run as training and leaving the remaining 20% unused during model parameter tuning to test the performance on unseen data. The numbered combinations of train-test splits are represented in Figure 3, and are comprised of five compact blocks of test set data of roughly four years covering the full range of observations. Cross-validation can provide a better idea about the stability of the models' performance given the relatively small amount of data available. Moreover, to study the intrinsic variability of the ANN models, they were also run five extra times on the same test set. The review of all of the model runs can be seen in Table 2.

The Welch's t-test was used to determine whether there is a significant difference in performance between the models from the five KGE values obtained for each of the cross-validation runs.

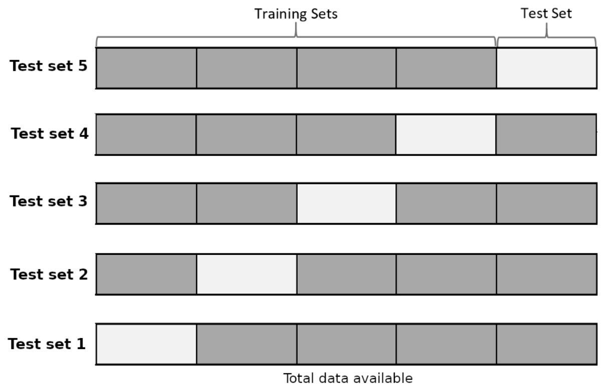

Figure 3: Representation of the five different test-train splits from the available data used to evaluate the models, with time increasing from left to right.

Table 2: Type and name of each model, whether or not it includes lagged variables and number of runs for every location using either cross-validation or test set 5 exclusively.

			Runs	
Model	Name	Lagged vars.	Cross-val.	Test set 5
MLR	mlr_no_lag	No	5x	-
	mlr_lag	Yes	5x	-
FCNN	fcn_no_lag	No	5x	5x
	fcn_lag	Yes	5x	5x
TCNN	tcn_lag	Yes	5x	5x

All of the code was run on Google Colab using Python 3.7.13. MLR was the fastest model by far, taking less than 10s for each run, even when including the lagged variables. FCNN runs took 3-4 min when not using lagged variables and 4-5 min when including them. When TCNN was run with the same hardware settings, it was roughly estimated to take up to 4h for each run, which made it unpractical, but using GPU boosting greatly reduced it to 6-12 min.

4 Results

The performances obtained from running the models are displayed in Figure 4. The box plots help visualize the results obtained by running the models with the five different train and test samples (see Section 3.3). The full results are available in Appendix B.

All of the error-correction models showed a clear improvement over the performance of PCR-GLOBWB in both analysed locations. In Basel, the non-lagged version of FCNN showcased a slightly better average performance and less fluctuation between test sets compared to MLR, both reducing the variability from the PCR-GLOBWB predictions greatly. In Lobith, FCNN also performed better than MLR on average but, on this occasion, the model fluctuations were higher, and there was not a clear improvement in variability compared to PCR-GLOBWB.

Including lagged variables slightly decreased the performance of the FCNN models in both locations, as can be seen in the second column of Figure 4. While the PCR-GLOBWB model data is the same, it is displayed again for two reasons; it makes it easier to compare against the lagged error-correction models, and also the test regions changed slightly due to the introduction of lagged variables. TCNN performed better than the lagged version of FCNN in both locations, but in Lobith the *fcn_no_lag* model had marginally better results. Nevertheless, all models failed to match the performance of the *mlr_lag* model.

The points that the box plots represent as outliers are only coincident between FCNN and TCNN in a given location, but do not seem to indicate any relevant feature of the data or the models.

According to Welch's t-test, the only significant differences in performance, i.e., with a p-value lower than 0.05, were between PCR-GLOBWB

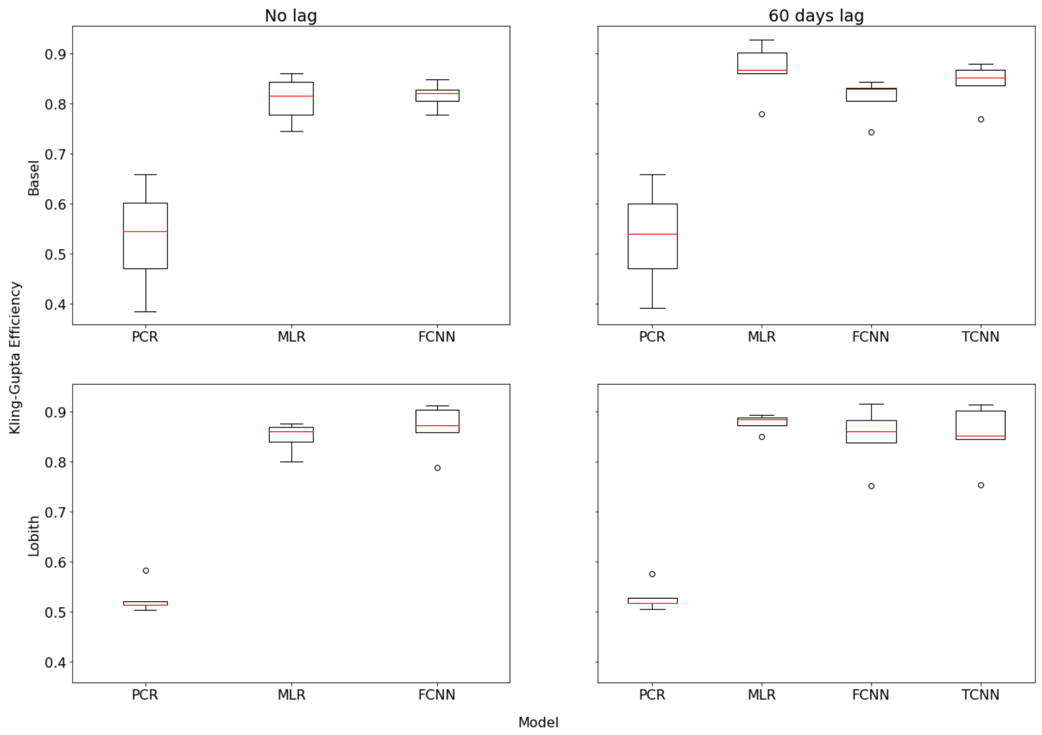

Figure 4: Distribution of KGE values for PCR-GLOBWB and the error-correction models without (left) and with (right) the addition of lagged variables, for Basel and Lobith.

and all of the error-correction models. This means that the difference between MLR, FCNN and TCNN, with or without the use of lagged variables, could not be considered large enough to extract any statistically significant conclusions.

The results in this project exceeded the KGE obtained by Y. Shen et al. (2022) using the same data, even though on that occasion the models were trained with only half the total extent of the data and they used a lag of 10. In Basel, their best-performing RF-based error-correction model only achieved a KGE of 0.75, while the models showcased in this project achieved average results ranging from 0.80 to 0.87. In Lobith, the improvement is not so much, 0.85 for RF compared to 0.85-0.88 in this project. As a reference, a sibling thesis project (Afshari Hemmatalikeykha, 2022), which used the same data and an ANN approach but predicted the streamflow directly from lagged meteorological variables only, obtained a lower KGE of 0.72 in Basel and 0.84 in Lobith. The same analysis without lagged variables yielded even worse results.

The streamflow forecast from all of the models for the last two years of data (corresponding to test set 5) are shown in Figures 5 and 6 for Basel and Lobith respectively. The observed values are displayed in black, the PCR-GLOBWB model prediction in blue and, for each plot, the different error-correction model forecasts are in red.

The predictions from all of the models showed a similar shape; they inherited a struggle to accurately predict high values of streamflow from PCR-GLOBWB, while, for low-flow regions, they fitted the observations much better. In general, PCR-GLOBWB overestimated the streamflow predictions, a tendency that is fixed by the error-correction models, which even underestimated a bit the results, especially in Lobith. It can be seen that the highest error in the predictions occurred in an unusual streamflow peak in early spring 1999 for both locations, which all models failed to properly predict, even though they showed some improvement compared to PCR-GLOBWB. It can also be noted that *mlr_lag*, the best performing model, was able to better capture the variability in observations, whereas the others only modified the PCR-GLOBWB values in a more generic way, missing the higher frequency patterns of the observations. Examples of this behaviour can be observed in the autumn of 1999 for Basel, and the late spring of 2000 for Lobith.

4.1 Model variability

Contrary to MLR, there was some variability in the performance of the ANN models when run on the same train and test set, mainly due to the parameter tuning, which relies on a small validation set, but also caused by their stochastic nature. The results, though, showed very similar performance within runs on the same test set.

For FCNN, the standard deviations of KGE values obtained by repeatedly running it on test set 5 were 0.025 and 0.011 (with and without lagged variables) for Basel, and 0.029 and 0.018 for Lobith. This was approximately half the amount obtained when running the model on the different test sets, 0.049 and 0.026 for Basel, and 0.062 and 0.040 for Lobith. For TCNN, this

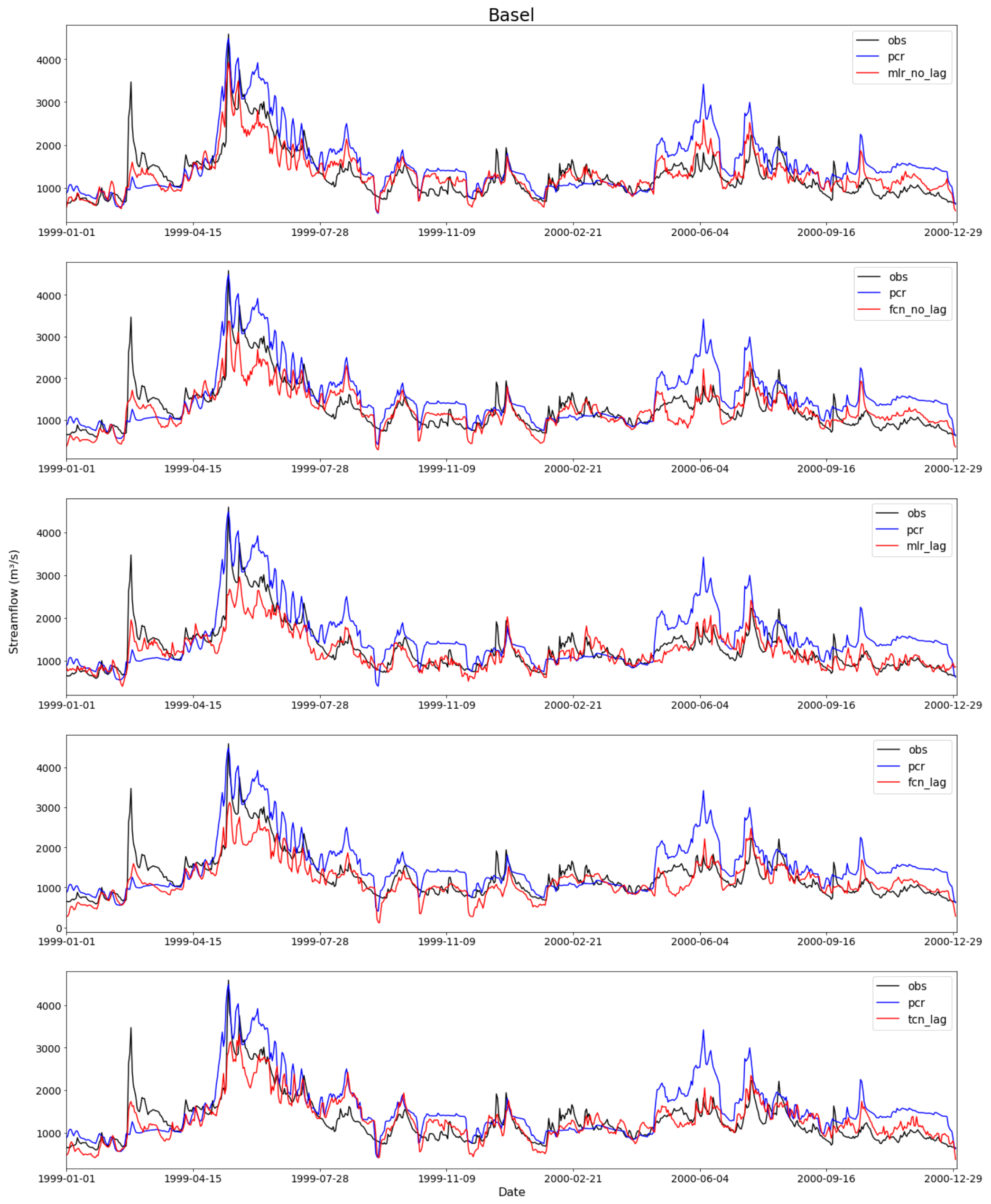

Figure 5: Streamflow observations (black), predictions from PCR-GLOBWB (blue) and from the different error-correction models (red) in Basel from the years 1999 and 2000.

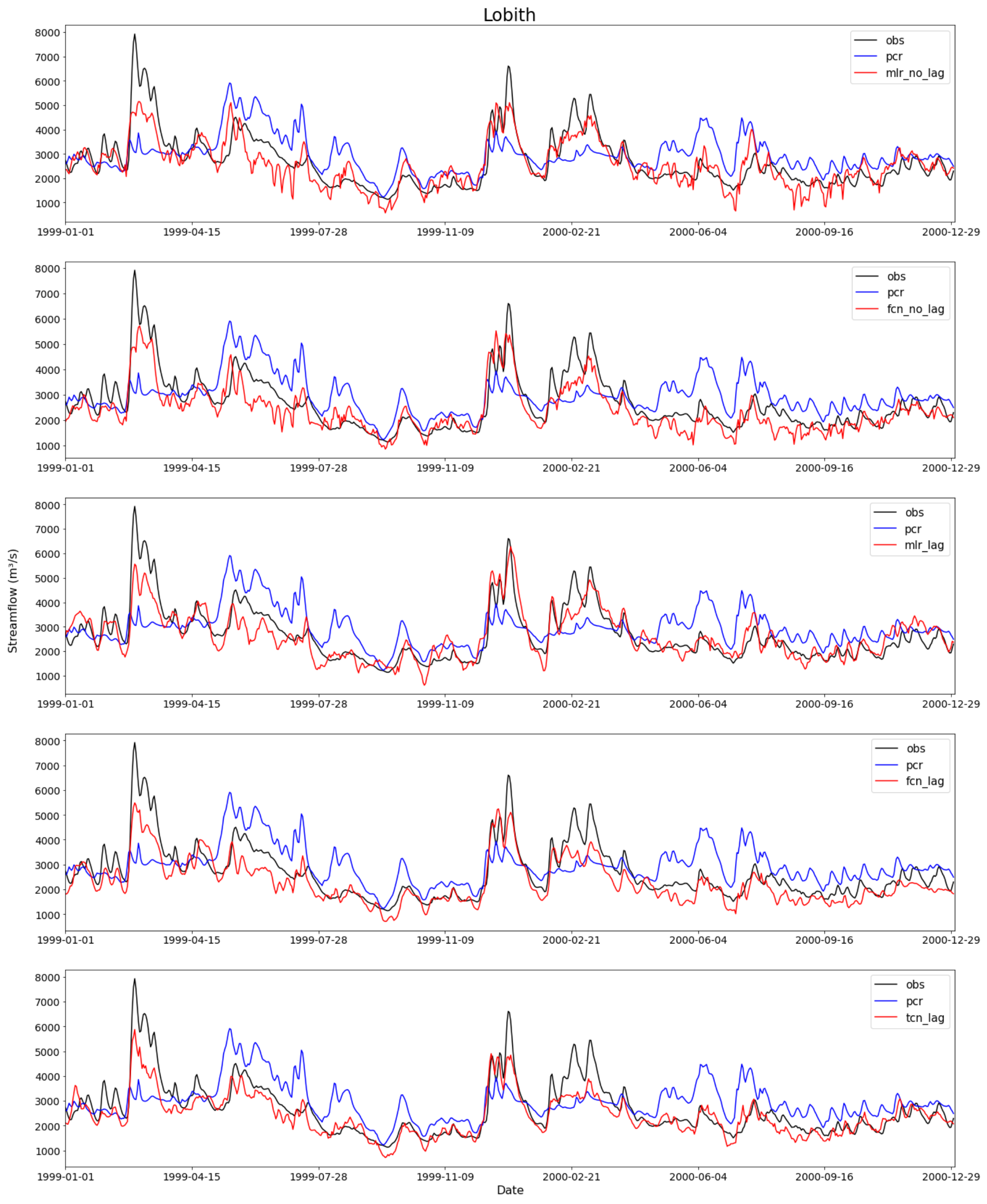

Figure 6: Streamflow observations (black), predictions from PCR-GLOBWB (blue) and from the different error-correction models (red) in Lobith from the years 1999 and 2000.

ratio was more than six times smaller, from 0.043 to 0.006 in Basel and from 0.063 to 0.010 on Lobith. Therefore, it can be concluded that the largest part of the variance came from using different train-test splits (especially from the test sets, as 3/4 of the train set was shared between any pair of runs) and not from the models' intrinsic variability.

It is also interesting to analyze the values of the tuned parameters for every run and their variation. They can be examined in Appendix B. For the non-lagged version of FCNN, the number of units for both layers oscillated between 200 and 175 consistently in the two locations. The lagged FCNN model still featured 200 and 175 as the most common number of units but, in Basel, there was more variability as the number of units dropped sporadically to much lower values, in some cases even when using the same test set. For the TCNN model, the *nb_filters* parameter also tended to reach the upper bound, either 200 or 175, showing perfect coherence between runs within the same test set for both locations.

4.2 Performance consistency

In order to analyze if the models performed constantly well for different observed values of streamflow, their cumulative frequency curves are plotted against the ones obtained from the observations in Figures 7 and 8, for Basel and Lobith respectively. In this plot, the full range of the available data is covered, made possible by cross-validation, and thus providing a more general picture of how the models perform.

A common behaviour for all models was to overestimate the amount of extremely low streamflow values, up to approximately 500 m^3/s in Basel and to 1000 m^3/s in Lobith, which were extremely rare in the observations. Surprisingly, all models seemed to perform worse than PCR-GLOBWB in that aspect, and only TCNN showed a distribution that closely resembled the observations reliably for both locations.

The bulk of the data, comprised between 500-1500 m^3/s in Basel and between 1000-3000 m^3/s in Lobith was not properly captured by the PCR-GLOBWB model, which clearly overestimated the predictions for this range of values. This was fixed by the rest of the models, as exemplified in Figures

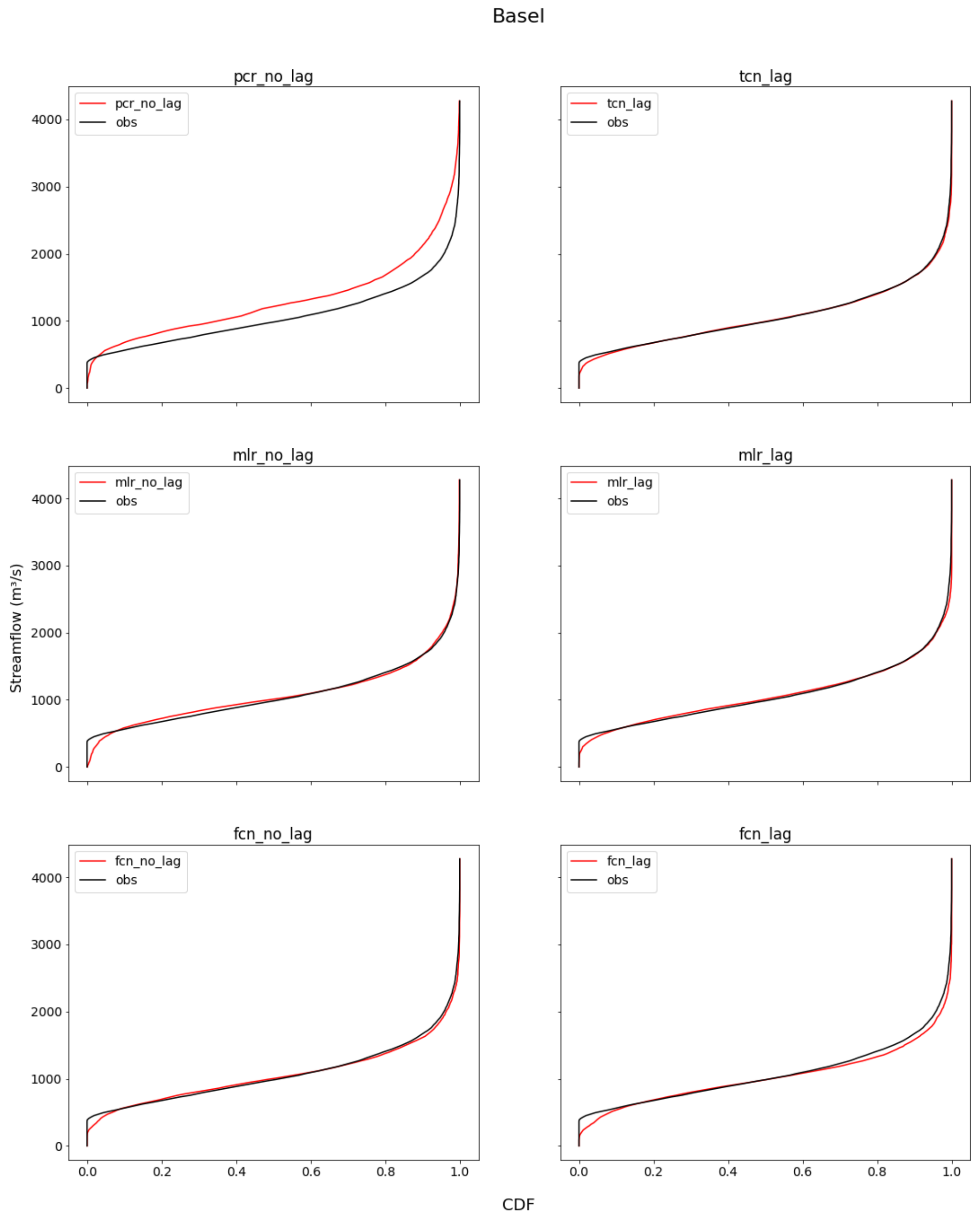

Figure 7: Cumulative frequency curve of the observations (black) compared to PCR-GLOBWB and the five error-correction models (red) in Basel.

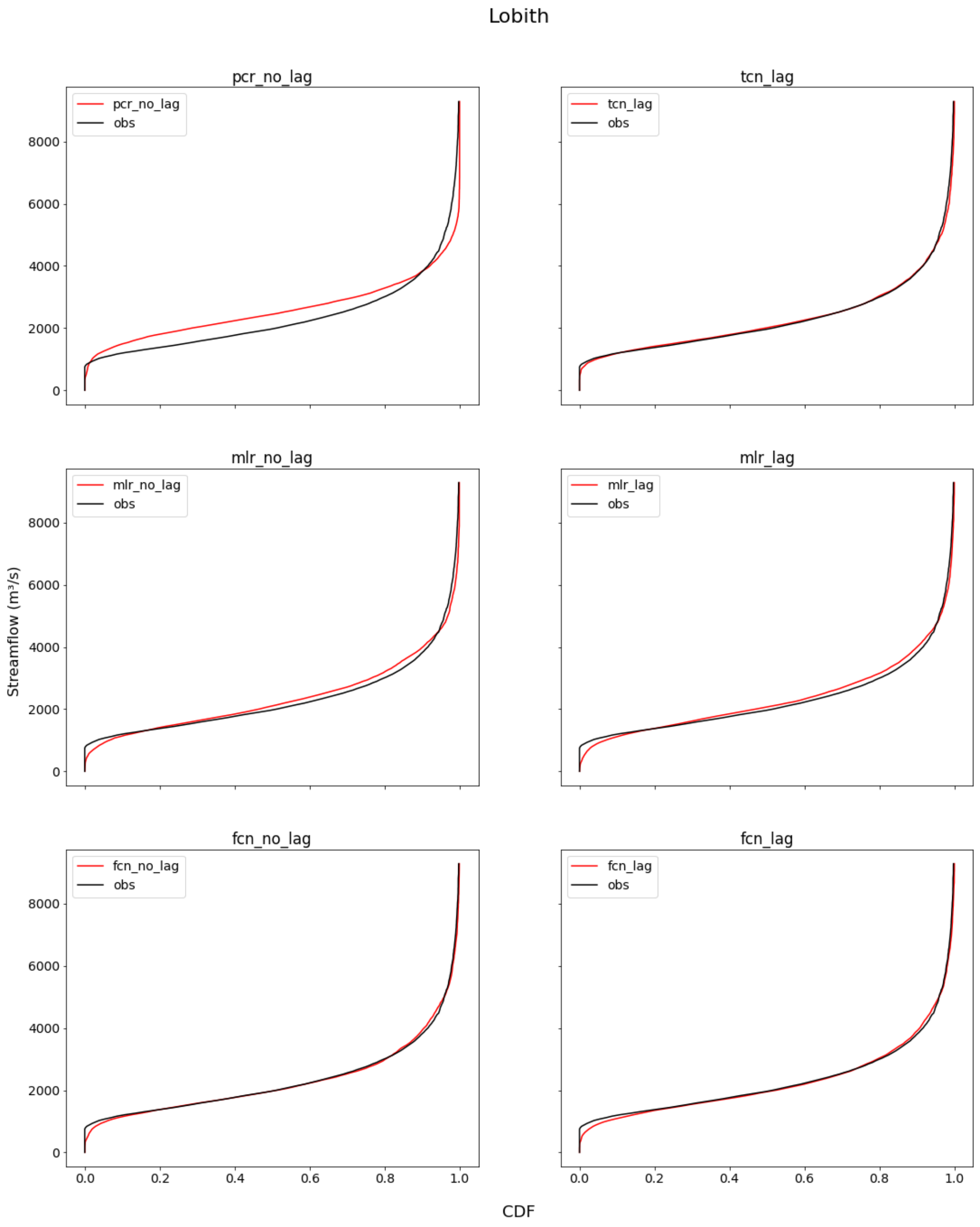

Figure 8: Cumulative frequency curve of the observations (black) compared to PCR-GLOBWB and the five error-correction models (red) in Lobith.

5 and 6. The MLR still suffered from this overestimation, although the lagged version fitted the observations much better. For the FCNN with and without lag, this only happened towards the higher portion of streamflow values while, for TCNN, the curve showed an almost perfect fit during the whole extent.

Finally, for the highest observed streamflow values, above 1500 m^3/s in Basel and above 3000 m^3/s in Lobith, there was a different behaviour between the two locations. In Basel, PCR-GLOBWB overestimated the presence of high streamflow values, while most of the models showcased a slightly lower curve at first but then became very similar to the observations for the highest values. TCNN, though, also fitted the curve very well in this range. In Lobith the opposite happens, as the prediction curve of the models was generally much steeper than the one from the observations in the higher part of the spectrum, indicating that they failed to predict the largest values. This was more noticeable in PCR-GLOBWB, but could be observed to an extent in the other models as well. On this occasion, both FCNN models had a better fit.

To further support this evidence from another point of view, plots of the observed against predicted streamflow values have been obtained and can be found in Appendix A. A few more characteristics of the predictions are also discussed there.

5 Conclusion and Discussion

The results clearly showed a remarkable increase in the performance of the error-correction models over PCR-GLOBWB in both locations, confirming the validity of this approach. It also obtained better results than the sibling project predicting streamflow directly from meteorological variables (Afshari Hemmatalikeykha, 2022), the difference being especially large in Basel, possibly due to the nival regime present in the area that would be better captured by the PCR-GLOBWB model.

However, the use of ANNs did not provide a significant change in performance compared to MLR. Despite this fact, assigning more observations to the training set improved the performance of all models compared to Y.

Shen et al. (2022), even with the linear model, so it would be interesting to analyze if it is possible to increase the performance even further by using a larger volume of data which may, in turn, favour the ANNs.

The addition of lagged variables up to 60 days did not increase the performance significantly, although the best-performing model was the lagged version of MLR. The main reason for this low increase in performance might be the fact that PCR-GLOBWB model states were used as predictors. These model state variables already capture some memory in the system, e.g. through the groundwater and snow storage. The TCNN model generally performed better than FCNN, which even decreases its performance slightly with the addition of lag, but the gain was not significant enough to draw any solid conclusions. On the other hand, of all of the models in both locations, TCNN showcased the most similar cumulative distribution of streamflow values compared to the observed one, especially in the lower values, which can be crucial as indicators of extreme drought.

Even though in this study the ANNs did not show a substantial enough improvement compared to linear regression that can justify their use, given their much higher complexity and computational expense, this should not be taken as proof to disregard the non-linearity of streamflow forecasting. Firstly, the use of PCR-GLOBWB state variables are already taking into account some of the non-linearity of the problem. In addition, ANNs have many hyperparameters, the most important ones described in Section 3, which can affect their performance greatly (Claesen & De Moor, 2015). Due to the scope of this project and the computational power available, hyperparameter tuning had to be severely limited. Both the units per layer in FCNN and the *nb_filters* in TCNN took values close to the upper bound for all of the model runs, which could indicate that higher values may provide better results. Additionally, a more comprehensive analysis of the effect of the other hyperparameters on the performance of the models would be desirable. The success of the linear models for this type of error-correction model, though, is undeniable and should be analyzed more in-depth in additional research.

Finally, the variation in performance by using a subset of input variables or by changing the lag would also be interesting to investigate further. Some preliminary results suggest that the performance could fluctuate substantially between seasons, which may also be explored. Ultimately, this work opens many more questions and challenges to be tackled in the future of hydrology.

Code availability

The data and source codes used in this study are available on https://github.com/oriol-pomarol/final_thesis_project.

References

Afshari Hemmatalikeykha, M. (2022). *Using LSTM and XGBoost for streamflow prediction based on meteorologial time series data* (Master's thesis). University of Utrecht. Utrecht.

Chollet, F. et al. (2015). Keras. https://keras.io

Claesen, M., & De Moor, B. (2015). Hyperparameter Search in Machine Learning [Number: arXiv:1502.02127 arXiv:1502.02127 [cs, stat]]. https://doi.org/10.48550/arXiv.1502.02127

Duan, S., Ullrich, P., & Shu, L. (2020). Using Convolutional Neural Networks for Streamflow Projection in California. *Frontiers in Water*, *2*, 28. https://doi.org/10.3389/frwa.2020.00028

Gao, C., Gemmer, M., Zeng, X., Liu, B., Su, B., & Wen, Y. (2010). Projected streamflow in the Huaihe River Basin (2010–2100) using artificial neural network. *Stochastic Environmental Research and Risk Assessment*, *24*(5), 685–697. https://doi.org/10.1007/s00477-009-0355-6

Gupta, H. V., Kling, H., Yilmaz, K. K., & Martinez, G. F. (2009). Decomposition of the mean squared error and NSE performance criteria: Implications for improving hydrological modelling. *Journal of Hydrology*, *377*(1), 80–91. https://doi.org/10.1016/j.jhydrol.2009.08.003

Hallouin, T. (2021). Hydroeval: An evaluator for streamflow time series in Python. https://doi.org/10.5281/zenodo.4709652

Isik, S., Kalin, L., Schoonover, J. E., Srivastava, P., & Graeme Lockaby, B. (2013). Modeling effects of changing land use/cover on daily streamflow: An Artificial Neural Network and curve number based hybrid approach. *Journal of Hydrology*, *485*, 103–112. https://doi.org/10.1016/j.jhydrol.2012.08.032

Lange, H., & Sippel, S. (2020). Machine Learning Applications in Hydrology. In D. F. Levia, D. E. Carlyle-Moses, S. Iida, B. Michalzik, K. Nanko, & A. Tischer (Eds.), *Forest-Water Interactions* (pp. 233–257). Springer International Publishing. https://doi.org/10.1007/978-3-030-26086-6_10

Maheswaran, R., & Khosa, R. (2012). Wavelet–Volterra coupled model for monthly stream flow forecasting. *Journal of Hydrology*, *450-451*, 320–335. https://doi.org/10.1016/j.jhydrol.2012.04.017

Nearing, G. S., Kratzert, F., Sampson, A. K., Pelissier, C. S., Klotz, D., Frame, J. M., Prieto, C., & Gupta, H. V. (2021). What Role Does Hydrological Science Play in the Age of Machine Learning? [_eprint:

https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2020WR028091]. *Water Resources Research*, *57*(3), e2020WR028091. https://doi.org/10.1029/2020WR028091

Noori, N., & Kalin, L. (2016). Coupling SWAT and ANN models for enhanced daily streamflow prediction. *Journal of Hydrology*, *533*, 141–151. https://doi.org/10.1016/j.jhydrol.2015.11.050

O'Malley, T., Bursztein, E., Long, J., Chollet, F., Jin, H., Invernizzi, L., et al. (2019). KerasTuner. https://github.com/keras-team/keras-tuner

Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., & Kavukcuoglu, K. (2016). WaveNet: A Generative Model for Raw Audio.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*(85), 2825–2830. Retrieved June 30, 2022, from http://jmlr.org/papers/v12/pedregosa11a.html

Rémy, P. (2022). Keras TCN [original-date: 2018-03-22T02:40:06Z]. Retrieved May 26, 2022, from https://github.com/philipperemy/keras-tcn

Shen, C., Chen, X., & Laloy, E. (2021). Editorial: Broadening the Use of Machine Learning in Hydrology. *Frontiers in Water*, *3*, 681023. https://doi.org/10.3389/frwa.2021.681023

Shen, Y. (2021). Co822ee/PCR-GLOBWB_error-correction: V2. https://doi.org/10.5281/zenodo.5068517

Shen, Y., Ruijsch, J., Lu, M., Sutanudjaja, E. H., & Karssenberg, D. (2022). Random forests-based error-correction of streamflow from a large-scale hydrological model: Using model state variables to estimate error terms. *Computers & Geosciences*, *159*, 105019. https://doi.org/10.1016/j.cageo.2021.105019

Sutanudjaja, E. (2017). PCR-GLOBWB_model: PCR-GLOBWB version v2.1.0_beta_1. https://doi.org/10.5281/zenodo.247139

A Appendix

Figures A1 and A2 showcase the observed against predictions plots for every model in Basel and Lobith respectively. It can be seen for which values of streamflow the models performed the best, and whether they overestimated (below 1:1 line) or underestimated (above 1:1 line) in their predictions.

The results reinforce the idea that all models but TCNN tended to predict unrealistically low values of streamflow. Also, the limitations of the PCR-GLOBWB model to adapt to the different regions became clear, overestimating the streamflow in Basel and failing to predict high streamflow values in Lobith, both correctly addressed by the error-correction models. Lastly, a general increase in prediction variability could be seen as the streamflow values increase.

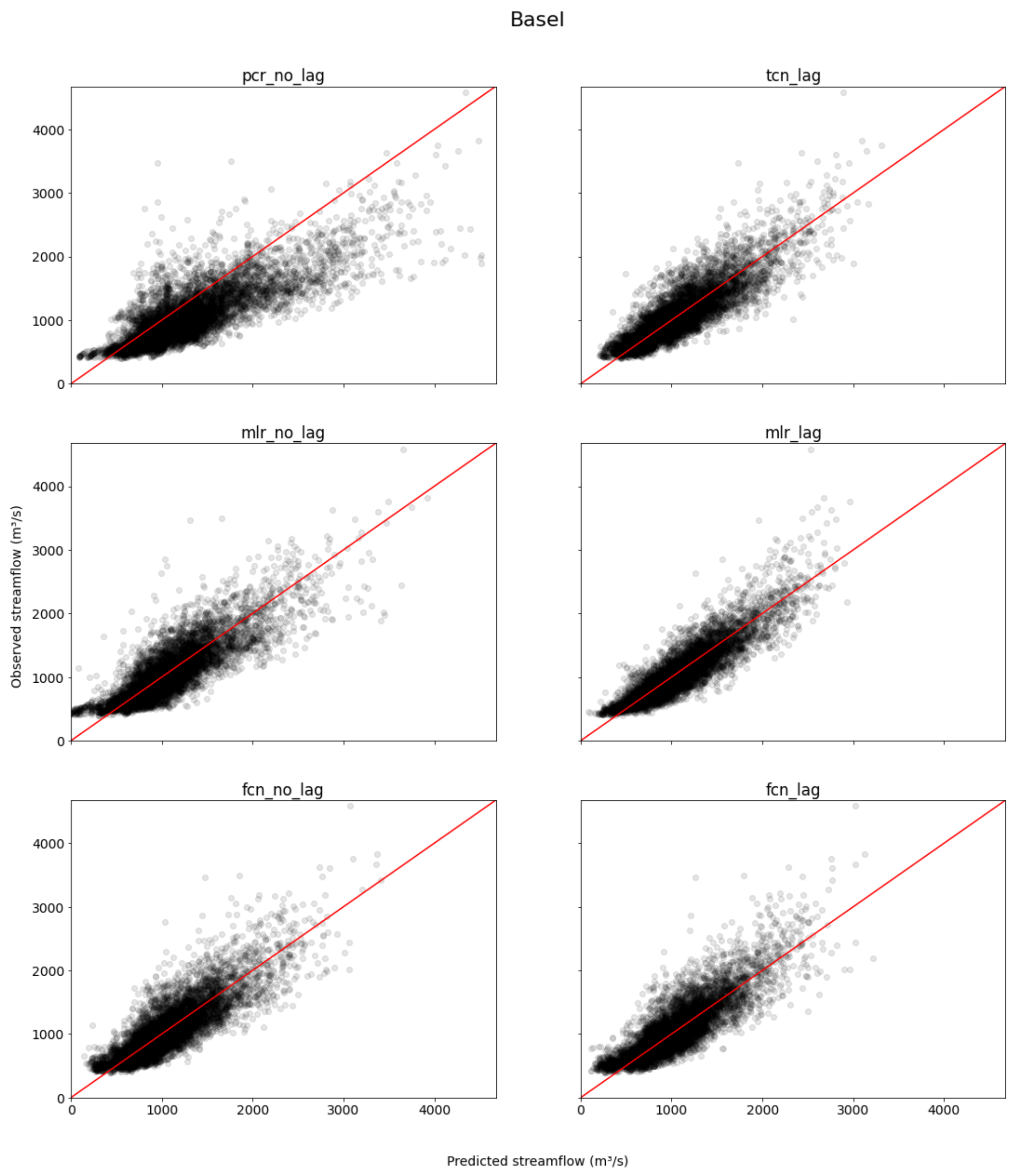

Figure A1: Plot of observed against predicted streamflow values for PCR-GLOBWB (*pcr_ no_ lag*) and the five error-correction models (Table 2) in Basel. In red is the ideal 1:1 line.

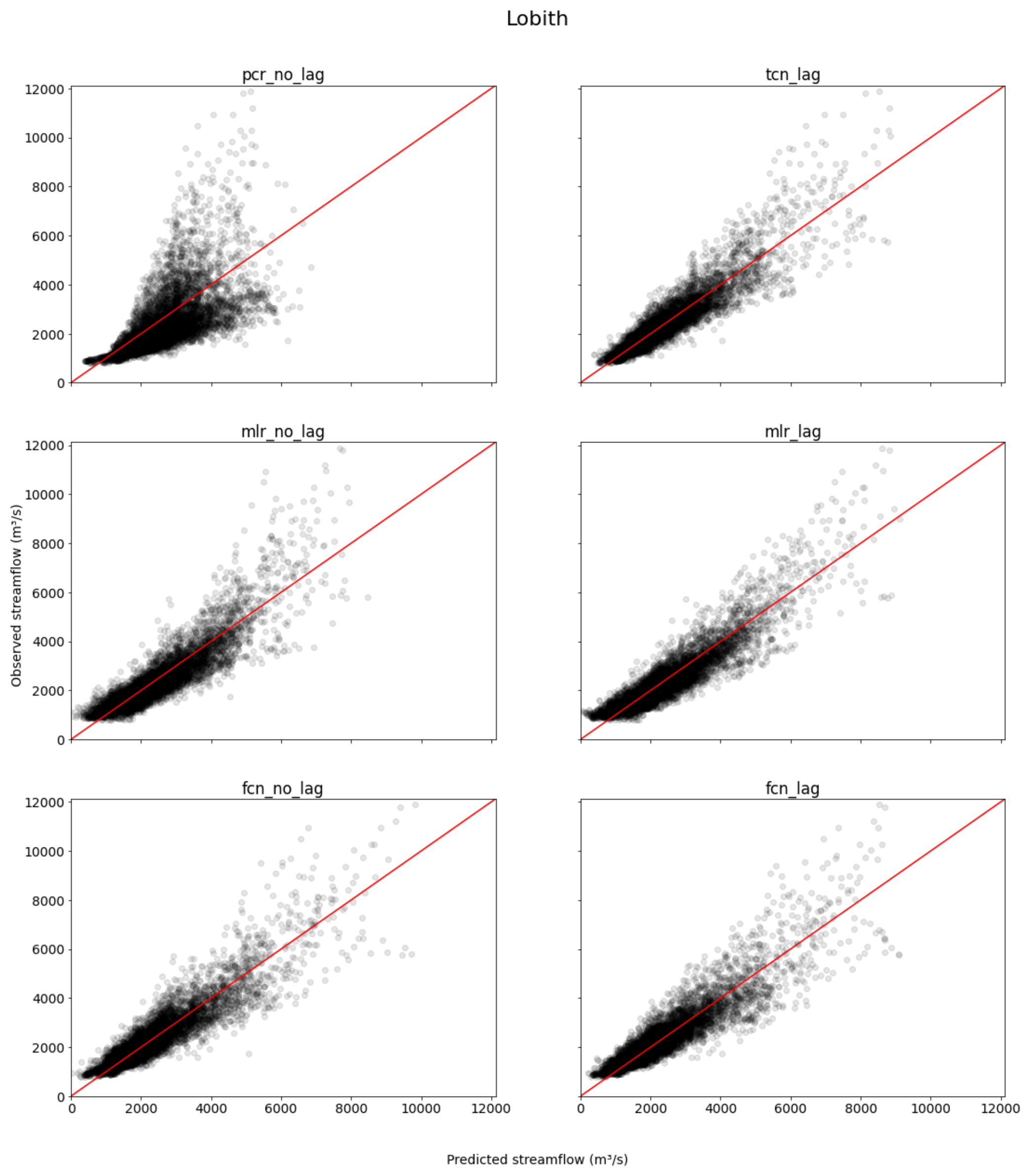

Figure A2: Plot of observed against predicted streamflow values for PCR-GLOBWB (*pcr_no_lag*) and the five error-correction models (Table 2) in Lobith. In red is the ideal 1:1 line.

B Appendix

In the tables available in this section, the performance of the models, including KGE and the unused NSE, as well as the value of the tuned parameters (if applicable) for every run, is displayed for both Basel and Lobith separated between the models that used lagged variables and the ones that did not.

Table B1: Results obtained from the *no_ lag* models in Basel.

Model	Parameters	Description	NSE	KGE
PCR	-	Test set 1	-0.35	0.39
		Test set 2	0.08	0.47
		Test set 3	-0.12	0.55
		Test set 4	0.11	0.60
		Test set 5	0.27	0.66
MLR	-	Test set 1	0.49	0.75
		Test set 2	0.69	0.84
		Test set 3	0.55	0.78
		Test set 4	0.72	0.86
		Test set 5	0.72	0.82
FCN	Units: 200, 175	Test set 1	0.58	0.78
	Units: 200, 175	Test set 2	0.72	0.83
	Units: 175, 200	Test set 3	0.60	0.81
	Units: 200, 200	Test set 4	0.73	0.85
	Units: 200, 175	Test set 5	0.73	0.82
	(Test set 5)			
	Units: 200, 200	Run 1	0.74	0.84
	Units: 200, 200	Run 2	0.73	0.84
	Units: 200, 200	Run 3	0.73	0.84
	Units: 175, 175	Run 4	0.72	0.82
	Units: 200, 175	Run 5	0.74	0.85

Table B2: Results obtained from the *lag* models in Basel.

Model	Parameters	Description	NSE	KGE
PCR	-	Test set 1	-0.35	0.39
		Test set 2	0.09	0.47
		Test set 3	-0.16	0.54
		Test set 4	0.11	0.60
		Test set 5	0.28	0.66
MLR	-	Test set 1	0.72	0.86
		Test set 2	0.85	0.93
		Test set 3	0.73	0.87
		Test set 4	0.80	0.90
		Test set 5	0.79	0.78
FCN	Units: 175, 200	Test set 1	0.55	0.74
	Units: 200, 100	Test set 2	0.77	0.84
	Units: 175, 200	Test set 3	0.66	0.83
	Units: 200, 200	Test set 4	0.67	0.83
	Units: 175, 150	Test set 5	0.72	0.80
	Test set: 5			
	Units: 125, 200	Run 1	0.73	0.82
	Units: 150, 200	Run 2	0.75	0.84
	Units: 200, 50	Run 3	0.72	0.81
	Units: 125, 200	Run 4	0.76	0.83
	Units: 175, 125	Run 5	0.70	0.79
TCNN	Units: 175	Test set 1	0.59	0.77
	Units: 200	Test set 2	0.78	0.88
	Units: 175	Test set 3	0.67	0.84
	Units: 200	Test set 4	0.73	0.85
	Units: 200	Test set 5	0.74	0.87
	Test set: 5			
	Units: 200	Run 1	0.76	0.87
	Units: 200	Run 2	0.74	0.87
	Units: 200	Run 3	0.76	0.87
	Units: 200	Run 4	0.74	0.86
	Units: 200	Run 5	0.74	0.86

Table B3: Results obtained from the *no_lag* models in Lobith.

Model	Parameters	Description	NSE	KGE
PCR	-	Test set 1	0.32	0.52
		Test set 2	0.22	0.51
		Test set 3	0.33	0.58
		Test set 4	0.40	0.51
		Test set 5	0.32	0.50
MLR	-	Test set 1	0.74	0.86
		Test set 2	0.80	0.87
		Test set 3	0.78	0.88
		Test set 4	0.79	0.80
		Test set 5	0.80	0.84
FCN	Units: 175, 200	Test set 1	0.74	0.86
	Units: 200, 200	Test set 2	0.84	0.91
	Units: 200, 200	Test set 3	0.82	0.87
	Units: 200, 150	Test set 4	0.88	0.90
	Units: 200, 175	Test set 5	0.79	0.79
	Test set 5			
	Units: 175, 200	Run 1	0.81	0.79
	Units: 175, 200	Run 2	0.76	0.76
	Units: 200, 200	Run 3	0.83	0.82
	Units: 200, 175	Run 4	0.78	0.77
	Units: 200, 175	Run 5	0.81	0.80

Table B4: Results obtained from the *lag* models in Lobith.

Model	Parameters	Description	NSE	KGE
PCR	-	Test set 1	0.32	0.53
		Test set 2	0.22	0.52
		Test set 3	0.34	0.58
		Test set 4	0.40	0.51
		Test set 5	0.32	0.51
MLR		Test set 1	0.78	0.87
		Test set 2	0.84	0.89
	-	Test set 3	0.74	0.85
		Test set 4	0.86	0.89
		Test set 5	0.84	0.89
FCN	Units: 200, 200	Test set 1	0.74	0.84
	Units: 200, 200	Test set 2	0.85	0.88
	Units: 200, 150	Test set 3	0.82	0.86
	Units: 200, 200	Test set 4	0.86	0.92
	Units: 200, 175	Test set 5	0.86	0.75
	Test set 5			
	Units: 175, 175	Run 1	0.78	0.78
	Units: 200, 175	Run 2	0.79	0.79
	Units: 200, 200	Run 3	0.69	0.72
	Units: 200, 175	Run 4	0.78	0.78
	Units: 200, 175	Run 5	0.74	0.76
TCNN	Units: 200	Test set 1	0.82	0.90
	Units: 175	Test set 2	0.88	0.85
	Units: 200	Test set 3	0.81	0.84
	Units: 200	Test set 4	0.86	0.91
	Units: 200	Test set 5	0.79	0.75
	Test set 5			
	Units: 200	Run 1	0.82	0.78
	Units: 200	Run 2	0.82	0.79
	Units: 200	Run 3	0.82	0.78
	Units: 200	Run 4	0.82	0.79
	Units: 200	Run 5	0.82	0.77