Team DA Collective

The "DA Collective" team used an ensemble-of-transfer-function-models approach. The Pastas python module (Collenteur et al., 2019) was used as the underlying transfer-function model engine and was wrapped within an ensemble-based data assimilation framework. The ensemble-based data assimilation captured transfer-function input parameter uncertainty as well as temporally-varying forcing uncertainty. The forcing uncertainty was captured in Julian day multiplier parameters that were then transferred to the forecast period so that any biases learned during the training period could be propagated to the forecast period. We also explicitly considered temporally correlated noise in the observed groundwater levels used for training. The iterative ensemble form of the Gauss-Levenburg-Marquardt solution was used to assimilate the observed groundwater levels. An ensemble of 100 realizations was used across 5 iterations for data assimilation; we note that an adaptive rejection filter was used each iteration to remove realizations with statistical outlier phi values. Observed groundwater levels from 1 Jan 2008 to 31 Dec 2012 were held back for verification at each site. More details related to the approach can be found in the workflow python script.

References

- Collenteur, R. A., Bakker, M., Caljé, R., Klop, S. A., and Schaars, F.: Pastas: Open Source Software for the Analysis of Groundwater Time Series, Groundwater, 57, 877–885, https://doi.org/10.1111/gwat.12925, 2019

Team Gardenia

GARDENIA is a BRGM program (Bureau de Recherches Géologiques et Minières - French Geological Survey [1]), using usual meteorological time-series (P, PET, T) to calculate either flow rates or groundwater levels, through a lumped hydrologic model with simplified hydraulic laws. Those laws are, for example, based on factors such as “Half-time rising”, “Half-time depletion” and “Equivalent storage coefficient” which are closely related to diffusivity and storativity, in complement of soil and aquifer budgets. For this challenge, GARDENIA was used with daily records, although the calculations can be made at a 5 minutes to monthly time step. GARDENIA could eventually take into account snowmelt, pumping wells and multiple reservoirs. GARDENIA is installed with a simplified user interface (in French). GARDENIA has been verified by comparing its results with those from numerical models made with MARTHE, another BRGM groundwater model [2]. Various NASH factors and graphical outputs allow the user to appreciate the quality of calibrations.

References

- [1] https://www.brgm.fr/fr/logiciel/

- [2] Organisation : BRGM Author : Dominique Thiéry Document reference : BRGM/RP-64500-FR Title : Validation du code de calcul GARDÉNIA par modélisations physiques comparatives Date : June 2021

Team HydroSight

HydroSight is a flexible groundwater timeseries analysis package that, amongst other features, allows transfer function noise (TFN) modelling using a reformulation of the von Asmuth et al. (2005) approximated likelihood function for irregularly sampled head observations (Peterson and Fulton, 2019; Peterson and Western, 2014; available at https://github.com/peterson-tim-j/HydroSight, version 1.41.4). Our analysis used a nonlinear partitioning of snowmelt with the Degree-Day Factor (DDF) method (Çallı et al., 2022; Martinec, 1960), where melt water infiltrates to a two-layer 1D soil moisture ordinary differential equation (Peterson and Fulton, 2019). The evaporation is partitioned firstly to the snow melt, the remaining evaporative potential then drives evaporation from the top soil moisture layer and any remaining potential drives evaporation from the lower soil moisture layer. The free drainage from the deepest soil layer provides an estimate of daily recharge. Importantly, to ensure plausible estimates of recharge are produced, the partitioning of precipitation was constrained using probabilistic estimates of the Budyko actual evapotranspiration (AET) for the climatology of the site (Greve et al., 2015). This was achieved by constraining the average total modelled AET to be within the 10^{th} to 90^{th} percentile range of AET from Budyko curve (Peterson and Fulton, 2019). The outcome of this is an estimate of head resulting from a physically plausible model.

The input daily PET at each site was estimated with an R package *Evapotranspiration* (Guo et al., 2016) with the methods provided in Table 1. The modelled recharge was then convolved with a reformulated Pearson type III distribution function, whereby the convolution is numerically integrated using Simpson's composite rule and analytical integration for the first and the last time steps to negative infinity (Peterson and Western, 2014). Each model has 13 parameters to calibrate: 2 parameters (DDF factor, T_{melt}) for the snowmelt module, 7 parameters (SMSC, $SMSC_{deep}$, k_{sat}, $k_{sat,deep}$, β, β_{deep}, γ) for the two-layer soil moisture storage module, and 3 parameters for the weighting function (A, b, n) and one for the exponential noise function (α). All model parameters were jointly calibrated to the observed heads using an estimate of the log-likelihood (Peterson and Western, 2014) and the global scheme called Shuffled Complex Evolution with principal components analysis–University of California at Irvine (SP-UCI) (Chu et al., 2011) with 4 complexes per model parameter. The percentage change allowed in the objective function before convergence is set to 1×10^{-6} and the calibration stops when at least 10 evolution loops meet the convergence criteria.

Table 1 The PET methods used for each site.

Sites	PET methods
Germany	Makkink (Tmax, Tmin, Rs)
Netherlands	Makkink (Tmax, Tmin, Rs)
Sweden1	Makkink (Tmax, Tmin, Rs)
Sweden2	Penman-Monteith (Tmax, Tmin, RHmax and RHmin, Rs, Wind speed)
USA	Hargreaves (T)

References

- Von Asmuth, J. R. and Bierkens, M. F. P.: Modeling irregularly spaced residual series as a continuous stochastic process, Water Resour. Res., 41(12), 1–11, doi:10.1029/2004WR003726, 2005.
- Çallı, S. S., Çallı, K. Ö., Tuğrul Yılmaz, M. and Çelik, M.: Contribution of the satellite-data driven snow routine to a karst hydrological model, J. Hydrol., 607(January), 127511, doi:10.1016/j.jhydrol.2022.127511, 2022.
- Chu, W., Gao, X. and Sorooshian, S.: A new evolutionary search strategy for global optimization of high-dimensional problems, Inf. Sci. (Ny)., 181(22), 4909–4927, doi:10.1016/j.ins.2011.06.024, 2011.
- Greve, P., Gudmundsson, L., Orlowsky, B. and Seneviratne, S. I.: Introducing a probabilistic Budyko framework, Geophys. Res. Lett., 42(7), 2261–2269, doi:10.1002/2015GL063449, 2015.

- Guo, D., Westra, S. and Maier, H. R.: An R package for modelling actual, potential and reference evapotranspiration, Environ. Model. Softw., 78, 216–224, doi:10.1016/j.envsoft.2015.12.019, 2016.
- Martinec, J.: The degree-day factor for snowmelt runoff forecasting, IUGG Gen. Assem. Helsinki, IAHS Comm. Surf. Waters, 51, 468–477, 1960.
- Peterson, T. J. and Fulton, S.: Joint estimation of gross recharge, groundwater usage, and hydraulic properties within HydroSight, Groundwater, 57(6), 860–876, doi:10.1111/gwat.12946, 2019.
- Peterson, T. J. and Western, A. W.: Nonlinear time-series modeling of unconfined groundwater head, Water Resour. Res., 50(10), 8330–8355, doi:10.1002/2013WR014800, 2014.

Team Janis

The model was developed using the R programming language (version 4.2.2., R Core Team, 2022) and the tidymodels (version 1.0.0) package environment (Kuhn and Wickham, 2020). The random forest algorithm from the "ranger" package (version 0.14.1, Wright and Ziegler, 2017) was utilized. To capture temporal dynamics of the groundwater system, the given descriptor features were summarized over various time periods: either averaged over the last 5, 30, 60 and 180 days or accumulated over the last 5, 30, 60, 180 and 270 days. Additional predictors such as accumulated positive daily mean temperatures and daily excess precipitation (rr minus et) were also used, while additional excess precipitation features were computed using accumulated 5, 30, 60, 180 and 270-day precipitation and evaporation data. In addition, two temporal features were included: day of the year and a month. The total number of descriptor features was 68 for all but USA case, where 52 features were used. The only tuned hyperparameter for the model was *mtry* – the number of variables randomly selected at each node, which was done by using 8 splits of 2-year long resamples, while number of trees was set to 500 and parameter *min_n* to 1.

References

- Kuhn, M. and Wickham, H. 2020. Tidymodels: a collection of packages for modeling and machine learning using tidyverse principles. https://www.tidymodels.org
- R Core Team. 2022 R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/
- Wright, M. N. and Ziegler, A. 2017. ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R. *Journal of Statistical Software*, *77*, 1–17. https://doi.org/10.18637/jss.v077.i01

Team Mirkwood

We used an ensemble of random forest models implemented in the R language using the 'tidymodels' framework (Kuhn and Wickham, 2020) and the 'ranger' implementation of random forest (Wright and Ziegler, 2017). This approach is derived from the one described in Di Ciacca et al. (2023) but with several modifications to fit the purpose of this challenge. First, the moving averages or sums (right aligned) were calculated for all variables with widths of 7, 30, 100, 365 days. Second, both the randomly selected predictor number and minimum node size were tuned. More specifically, 50 combinations of randomly sampled values, between one and the number of variables for the predictor number, and between two and 40 for the minimum node size, were generated. Each combination was tested by training a random forest on 75% (time split) of the calibration dataset and tested on the remaining 25%, using RMSE as a metric. The best performing combination was used for the predictions. For uncertainty quantification, the 10 best performing combinations were repeated 10 times each to generate 100 realizations. These 100 realizations were used to generate distributions of predicted values, which were then used to derive the prediction intervals. As seen in the results, this uncertainty quantification approach did not work well. Perhaps a better approach would be to use quantile regression forest (Meinshausen and Ridgeway, 2006).

References

- Di Ciacca, A., Wilson, S., Kang, J., Wöhling, T., 2023. Deriving transmission losses in ephemeral rivers using satellite imagery and machine learning. Hydrol. Earth Syst. Sci. 27, 703–722. https://doi.org/10.5194/hess-27-703-2023
- Meinshausen, N., Ridgeway, G., 2006. Quantile regression forests. J. Mach. Learn. Res. 7.

Team MxNI

Team MxNl employed a model ensemble consisting of individually optimized members for each of the five locations. The selection of ensemble members was drawn from a common pool of candidate models, with automatic optimization conducted for each specific location. The candidate ensemble members comprised four shallow, non-sequential learners without a built-in sense of time step order: the Multi-Layer Perceptron (MLP) (Venables and Ripley, 2002), Random Forest (RF) (Wright and Ziegler, 2017), Radial Basis Function support Vector Machine (RBF-SVM) (Karatzoglou et al., 2004), and Polynomial Support Vector Machine (P-SVM) (Karatzoglou et al., 2004). For the generation of different hyperparameter variants of each basic learner, a simple grid search with five different values or levels for each parameter was used. The default value ranges for all hyperparameters as implemented in the R package 'tune' (part of 'tidymodels' meta-package) were used (Kuhn, 2023). This leads to nnhyperparameters levels = 5nhyperparameters different variants per basic learner. Each of these variants is trained and validated using the same validation strategy. After an initial data split of 90% for tuning and 10% for testing the model ensemble once the members are selected from all candidates, a time series cross-validation with 50% in the first fold and 2 years length in all other folds was used. The RMSE was used to quantify the model performance during the tuning. From all the validated variants of basic learners, the two best performing variants were chosen as possible candidates for building the ensemble. The optimal combination of these candidate ensemble members was determined by optimizing the weights assigned to each candidate. The resulting model ensembles for each location are presented in Table 1. After identifying the optimal combination of weighted members, the model ensemble was retrained using the complete time series length. All modeling and processing were conducted using the R programming language, primarily utilizing the packages 'tidymodels', 'targets', and 'tidyverse' (Kuhn and Wickham, 2020; Landau, 2021; Wickham et al., 2019). The aim of this ensemble approach including the feature engineering was also automatising the modelling process across different locations.

References

- Karatzoglou, A., Smola, A., Hornik, K., and Zeileis, A. (2004). kernlab – an S4 package for kernel methods in R. Journal of Statistical Software, 11(9):1–20.
- Kuhn, M. (2023). tune: Tidy Tuning Tools. https://tune.tidymodels.org/, https://github.com/tidymodels/tune.
- Kuhn, M. and Wickham, H. (2020). Tidymodels: a collection of packages for modeling and machine learning using tidyverse principles.
- Landau, W. M. (2021). The targets r package: a dynamic make-like function-oriented pipeline toolkit for reproducibility and high-performance computing. Journal of Open Source Software, 6(57):2959.
- Venables, W. N. and Ripley, B. D. (2002). Modern Applied Statistics with S. Springer, New York, fourth edition. ISBN 0-387-95457-0.
- Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., Fran ̧cois, R.,Grolemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T. L., Miller, E., Bache, S. M., M ̈uller, K., Ooms, J., Robinson, D., Seidel, D. P., Spinu, V., Takahashi, K., Vaughan, D., Wilke, C., Woo, K., and Yutani, H. (2019). Welcome to the tidyverse. Journal of Open Source Software, 4(43):1686.
  Wright, M. N. and Ziegler, A. (2017). ranger : A fast implementation of random forests for high dimensional data in R and C++. Journal of Statistical Software, 77(1)

Team Selina_Yang

We built several supervised machine learning models, ranging from linear regression to non-linear models, and tested the performance of each one. As a result, the Support Vector Machine Regression model (SVR) performed the best among all the models in the Netherlands and Germany datasets. During exploratory data analysis (EDA), we discovered a very high correlation between mean temperature and minimum temperature (0.95) as well as between mean temperature and maximum temperature (0.98). Furthermore, mean global radiation exhibited a correlation coefficient of 0.99 with potential evaporation. Therefore, we removed minimum temperature, maximum temperature, and potential evaporation from the dataset since their absence did not affect the overall model performance. Additionally, we included lag features in the dataset during the data engineering phase. These lag features consisted of moving average values for the past 15, 30, and 90 days. To enable the model to recognize periodic patterns over time, we also introduced the day value feature. The final number of features in the dataset was 25. We employed the time series K-fold method to split the data into the training set, validation set, and test set. After completing the machine learning pipeline, we trained the model using five different random states. In each random state, we iterated over various hyperparameter combinations for the model. The models we experimented with during development included Lasso, SVR, and Random Forest. Based on the results, we selected the SVR model as the best model. Although it had a similar mean Mean Squared Error (MSE) as the random forest model, its MSE exhibited a lower standard deviation. Additionally, training an SVR model took approximately half the time required to train a random forest model. The model was implemented using the sklearn packaged in Python (https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html).

Team GEUS

We applied a Long Short Term Memory (LSTM) model for predicting the water level timeseries using keras functionalities in python. We trained well specific models with a single LSTM layer, using the mean-squared-error as loss functions for the central prediction and the confidence intervals (upper and lower boundaries) were trained separately using a quantile regression loss function (Koch et al. 2021). The LSTM models were designed to simulate a single day of water level given a sequence historic meteorological variables. The data processing and model development contained the following steps: 1) Interpolate all missing days, between the first and the last observation in the observed water level head timeseries using linear interpolation. 2) Process additional meteorological variables based on the provided data: Rolling window sums of rainfall rate (180, 365, 730 and 1095 days), rolling window sums of net rainfall rate (180, 365, 730 and 1095 days) and snow storage, snow melt and 90 day rolling window sum of snow melt. Snow storage and snow melt are implemented by applying the degree-day method as implemented in MIKE SHE (DHI, 2020), with melting temperature of 0 C and maximum wet snow fraction of 0 and degree-day melting coefficient of 3 mm/C/d. 3) Calibration of LSTM parameters individually for each well: dropout, recurrent_dropout, learning_rate, n_steps (input sequence length), batchsize, n_cells, as well as the window size of the rolling window sums of the supplementary meteorological variables (or the omission of those). For calibration we used the Pareto Archived Dynamically Dimensioned Search (PADDS) in the OSTRICH optimization software (Matott, 2017; Asadzadeh and Tolson, 2013). The first four years of each groundwater head timeseries served as test data, the remainder as training data and the Kling Gupta Efficiency was used as performance metric. 4) Apply optimized parameters to model the entire period required for the submission.

References

- Asadzadeh, M. and Tolson, B.: Pareto archived dynamically dimensioned search with hypervolume-based selection for multi-objective optimization, 45, 1489–1509, https://doi.org/10.1080/0305215X.2012.748046, 2013.
- DHI: MIKE SHE - User Guide and Reference Manual, https://manuals.mikepoweredbydhi.help/2020/Water_Resources/MIKE_SHE_Print.pdf, 2020.
- Koch, J., Gotfredsen, J., Schneider, R., Troldborg, L., Stisen, S., & Henriksen, H. J. (2021). High Resolution Water Table Modeling of the Shallow Groundwater Using a Knowledge-Guided Gradient Boosting Decision Tree Model. Frontiers in Water, 81.
- Matott, L. S.: OSTRICH – An Optimization Software Toolkit for Research Involving Computational Heuristics. Documentation and User's Guide. Version 17.12.19, http://www.civil.uwaterloo.ca/envmodelling/Ostrich.html, 2017.

Team LUHG

Team LUHG used the N-HiTS (Neural Hierarchical Interpolation for Time Series Forecasting; Challu et al., 2022) architecture through its implementation in the Darts (Herzen et al., 2022) Python forecasting library. N-HiTS extends functionality of the Neural Basis Expansion Analysis (N-BEATS; Oreshkin et al. 2020) method by leveraging multi-rate input sampling and hierarchical interpolation of forecasts, thereby enabling computationally efficient long-horizon time series forecasting. N-HiTS models were trained for each location to provide location-specific forecasts. In addition to the provided datasets, datasets of cumulative precipitation (15, 30 and 60 days) were generated for all locations. Specifically for the *USA* location, an additional dataset was generated, which includes the 30-day percentage change in river stage. Before training, the datasets were divided into an 80:20 ratio for training and validation purposes and then scaled (min-max) using the training portion to avoid information leakage. Hyperparameter settings were evaluated manually and *early stopping* was applied to reduce risk of overfitting. The Darts implementation of N-HiTS supports probabilistic forecasts by utilizing likelihood models. For each location, 200 samples were drawn using the Gaussian likelihood model to provide probabilistic forecasts.

References

- Challu, C., Olivares, K. G., Oreshkin, B. N., Garza, F., Mergenthaler-Canseco, M., & Dubrawski, A. (2022). N-HiTS: Neural Hierarchical Interpolation for Time Series Forecasting. *arXiv preprint arXiv:2201.12886.* https://doi.org/10.48550/arXiv.2201.12886
- Herzen, J., Lässig, F., Piazzetta, S. G., Neuer, T., Tafti, L., Raille, G., Van Pottelbergh, T., Pasieka, M., Skrodzki, A., Huguenin, N., Dumonal, M., Kościsz, J., Bader, D., Gusset, F., Benheddi, M., Williamson, C., Kosinski, M., Petrik, M., & Grosch, G. (2022). Darts: User-Friendly Modern Machine Learning for Time Series. *Journal of Machine Learning Research*, 23(124), 5442–5447.
- Oreshkin, B. N., Carpov, D., Chapados, N., & Bengio, Y. (2020). N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *8th International Conference on Learning Representations, ICLR 2020*.

Team M2C_BRGM

We utilized the Boundary Corrected-Maximal Overlap Wavelet Transform Deep Learning (BC-MODWT-DL) models (Chidepudi et al., 2023) for all four wells: Germany, Netherlands, Sweden_2, and the USA. All the given data for each well is considered as input variables. Three different DL models, namely LSTM, GRU, and BILSTM, were tested independently. The last 20% of the given data were used for testing, while the remaining 80% were used for training and validation of the models. Model parameters were optimized using Bayesian optimization. The final model was selected based on its performance on the test set. The GRU model exhibited better performance for Sweden_2, whereas LSTM performed better for the other wells. The wavelet transform pre-processing helps in extracting crucial time-frequency information from the input variables, allowing for the incorporation of multiscale changes. However, caution must be exercised when using this technique as a preprocessing step, as incorrect usage may lead to overestimation of results. (Chidepudi et al., 2023) provided a detailed description of the preprocessing and model development approach used in this framework where it is further demonstrated that the utilization of BC-MODWT as a preprocessing method enhances the simulation of groundwater levels affected by low-frequency variability, specifically ranging from multi-annual to decadal timescales. Nevertheless, the improvement was minimal in the case of groundwater levels dominated by annual variability.

References

- Chidepudi, S. K. R., Massei, N., Jardani, A., Henriot, A., Allier, D., & Baulon, L. (2023). A wavelet-assisted deep learning approach for simulating groundwater levels affected by low-frequency variability. Science of the Total Environment, 865(November 2022), 161035. https://doi.org/10.1016/j.scitotenv.2022.161035

Team TUD

A Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber 1997) model was used, as implemented in the Python packages TensorFlow v2.8.0(TensorFlow2022a)andKeras v2.8.0 (Chollet et al. 2015). The LSTM type of recurrent neural networks (RNN) is especially popular for simulating and predicting time series (e.g., Hewamalage, Bergmeir, and Bandara 2021). A simple and generic approach to creating the LSTM-RNN model was used (TensorFlow 2022b). With an emphasis on reducing model development time and assessing generic LSTM-RNN model performance, such a simple structure was chosen. Features were selected based on their covariance with hydraulic heads and normalized afterwards. A 10 % input-dropout, two 64-unit LSTM layers and a single-unit dense layer were used. The LSTM and dense layers were used with linear activation functions. While the past 50 values were used to predict the current value for both cases to ensure comparability, 5 (precipitation, mean temperature, pressure, humidity, and potential evaporation) and 3 (maximum temperature, river water stage, and precipitation) features were used for the Netherlands and USA cases, respectively. Both models had 51,009 parameters that were optimized during training as well as structure-related hyperparameters (e.g., number of past time-steps, number of units and layers), which were manually chosen. For each case, an ensemble of 100 models was trained, differing in the dropout. Ensemble outputs were then used to compute 95 % confidence intervals. The estimation of uncertainty therefore integrated the effect of random dropout and the resulting variation during training. The model was initially developed for the Netherlands case and subsequently applied to the USA case, only changing relevant features to emphasize the generic structure of the model. The models were set up and calibrated on a personal computer, increasing calibration time but reflecting a generally typical availability of resources.

References

- Chollet, François et al. (2015). Keras. url: https://keras.io.
- Hewamalage, Hansika, Christoph Bergmeir, and Kasun Bandara (Jan. 2021). “Recurrent Neural Networks for Time Series Forecasting: Current status and future directions”. en. In: International Journal of Forecasting 37.1, pp. 388–427. issn: 01692070. doi: 10.1016/j.ijforecast.2020.06.008. url: https://linkinghub.elsevier.com/retrieve/pii/ S0169207020300996 (visited on 06/28/2023).
- Hochreiter, Sepp and Jürgen Schmidhuber (Nov. 1997). “Long short-term memory”. In: Neural Computation 9.8, pp. 1735–1780. issn: 0899-7667. doi: 10.1162/neco.1997.9.8.1735. url: https://doi.org/10.1162/neco.1997.9.8.1735.
- TensorFlow, Developers (Feb. 2022a). TensorFlow. url: https://doi.org/10.5281/zenodo.5949125.
- (2022b).Timeseriesforecasting.url:https://www.tensorflow.org/tutorials/structured_data/time_series (visited on 12/20/2022).

Team RouhaniEtAl

We used Convolutional Neural Networks (CNNs) model developed by Wunsch et al. (2022) [1]. The CNNs model utilized in this work includes a 1-D convolutional layer with a fixed kernel size (three) and an optimal number of filters, followed by a Max-Pooling layer and a Monte-Carlo dropout layer with a fixed dropout of 50% to prevent overfitting. This high dropout rate necessitates solid training for the model. Following that is a thick layer with an optimal number of neurons, followed by a single output neuron. The Adam optimizer was used for a maximum of 100 training epochs with an initial learning rate of 0.001, and gradient clipping was utilized to prevent exploding gradients. Another regularization strategy that was considered to prevent the model from overfitting the training data was early halting with a patience of 15 epochs. Bayesian optimization was used to tune several model hyperparameters (HP) [2]: training batch size (16-256); input sequence length (1-365 for daily datasets) & (1-52 for weekly datasets); the number of filters in the 1D-Conv layer (1-256); and the size of the first dense layer (1-256). All models were built with Python 3.8 [3], the TensorFlow deep-learning framework [4], and its Keras API [5]. NumPy [6], Pandas [7], [8], Scikit-Learn [9], BayesOpt [10], Matplotlib [11], UnumPy [12] libraries were also utilized. [1].

Model workflow to reproduce

We use the parameters provided for each time series data to train the model and divide each time series into four parts to identify the optimum model configuration: training set, validation set, optimization set, and test set. The test set always uses the most recent four years of data provided. The first 80% of the remaining time series were utilized for training, the next 20% for early stopping (validation set), and the remaining 10% for testing during HP tuning (optimization set), each using 10% of the remaining time series. We used a maximum optimization step number of 150 for each model or stopped after 15 steps without improvement if a minimum of 60 steps was reached. To lessen reliance on the random number generator seed, we scaled the data to [-1,1] and employed an ensemble of 10 pseudo-randomly started models. We used Monte-Carlo dropout during simulation to estimate model uncertainty from 100 realizations for each of the ten ensemble members. Using 1.96 times the standard deviation of the resulting distribution for each time step, we calculated the 95% confidence interval from these 100 realizations. To assess simulation accuracy, we measured NSE, squared Pearson r (R^2), absolute and relative root mean squared error (RMSE/rRMSE), and absolute and relative Bias (Bias/rBias). We calculate NSE using a long-term mean of groundwater level before the test set rather than the test set mean value [13].

References

- [1] A. Wunsch, T. Liesch, and S. Broda, “Deep learning shows declining groundwater levels in Germany until 2100 due to climate change,” Nature Communications 2022 13:1, vol. 13, no. 1, pp. 1–13, Mar. 2022, doi: 10.1038/s41467-022-28770-2.
- [2] F Nogueira, “Bayesian Optimization: Open source constrained global optimization tool for Python,” 2014. https://github.com/fmfn/BayesianOptimization (accessed Jan. 12, 2023).
- [3] “Python Tutorial Release 3.8.1 Guido van Rossum and the Python development team,” 2020.
- [4] M. Abadi et al., “TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems,” Mar. 2016, doi: 10.48550/arxiv.1603.04467.
- [5] F. Chollet, “keras,” 2015. https://github.com/fchollet/keras (accessed Jan. 12, 2023).
- [6] S. van der Walt, S. C. Colbert, and G. Varoquaux, “The NumPy array: A structure for efficient numerical computation,” Comput Sci Eng, vol. 13, no. 2, pp. 22–30, Mar. 2011, doi: 10.1109/MCSE.2011.37.
- [7] W. Mckinney, “Data Structures for Statistical Computing in Python,” 2010.
- [8] T. pandas development team, “pandas-dev/pandas: Pandas,” Nov. 2022, doi: 10.5281/ZENODO.7344967.

- [9] F. Pedregosa FABIANPEDREGOSA et al., “Scikit-learn: Machine Learning in Python Gaël Varoquaux Bertrand Thirion Vincent Dubourg Alexandre Passos PEDREGOSA, VAROQUAUX, GRAMFORT ET AL. Matthieu Perrot,” Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011, Accessed: Jan. 12, 2023. [Online]. Available: http://scikit-learn.sourceforge.net.
- [10] “fmfn/BayesianOptimization: A Python implementation of global optimization with gaussian processes.” https://github.com/fmfn/BayesianOptimization (accessed Jan. 12, 2023).
- [11] J. D. Hunter, “Matplotlib: A 2D Graphics Environment,” Comput Sci Eng, vol. 9, no. 03, pp. 90–95, May 2007, doi: 10.1109/MCSE.2007.55.
- [12] “Welcome to the uncertainties package — uncertainties Python package 3.0.1 documentation.” https://pythonhosted.org/uncertainties/ (accessed Jan. 12, 2023).
- [13] A. Wunsch, T. Liesch, and S. Broda, “Groundwater level forecasting with artificial neural networks: A comparison of long short-term memory (LSTM), convolutional neural networks (CNNs), and non-linear autoregressive networks with exogenous input (NARX),” Hydrol Earth Syst Sci, vol. 25, no. 3, pp. 1671–1687, Apr. 2021, doi: 10.5194/HESS-25-1671-2021.

Team TUV

The Transformer is a sequence-to-sequence neural network, which was originally implemented for natural language processing by Vaswani et al. (2017). For the time series version, we implemented an encoder-only Transformer. Unlike recurrent neural networks, the Transformer processes information globally over the input sequence opposed to sequentially. The Transformer uses an attention mechanism to capture dependencies between all values in the input sequence, by computing scaled dot-products. The Transformer consists of an initial dense layer for embedding to increase the input dimensions. A positional encoding through sine and cosine function is added to the embedded data to store information about the temporal order of the input sequence. Next, the scaled dot products are calculated in the attention layer. Residual connections are utilized to save the data prior to and after the attention layer. Furthermore, layer normalization is implemented to normalize the interim outputs. Next, two 1D convolutional layers are used and residual connections are applied again. Finally, the resulting data is passed through a dense layer to create the daily forecast. We used all features to train the model and performed the training and tuning individually for every study site. To define the input sequence length, we chose a window size of 30 days. The number of model parameters ranges from 6189 to 10781, depending on the study site. To quantify the model uncertainty, we utilized Monte Carlo dropout.

References

- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.

Team Haidro

No description provided, see scripts.

Team UW

In this prediction, we used an RNN model based on LSTM units. We chose this model because our data is time-series data, and it is one of the most commonly used models in this field. When using this type of machine learning model, we usually have several hyperparameters to tune, including the length of the sequence for each training sample, the learning rate of the optimizer, the number of hidden units, and the batch size for each optimizer step. As you can see, I have tuned these hyperparameters in the file "model_pytorch_tuning.ipynb". There are some other hyperparameters that we decided to set manually, such as 1 for the number of layers and 1000 for the number of epochs. After hyperparameter tuning, we used 90 days as the length of the sequence, 0.01 as the learning rate, 20 for the number of hidden units, and 512 for the batch size. You can see the plots at the end of the tuning file for each hyperparameter. Usually, we try to select the set of hyperparameters with the lowest error in both training and validation, considering the fact that our model should not overfit (usually, if the error is low in the training set but high in the validation set, we have an overfitted model). The model was trained on an NVIDIA GeForce RTX 3070 Laptop GPU. As mentioned in the readme file, it takes about 5 hours for each location to tune the hyperparameters and train the model. We haven't used any hydrology-related Python packages in our code. (There are some packages like neural hydrology with tuned hyperparameters.) We have only used fundamental packages such as PyTorch, pandas, and scikit-learn.