Multi-Horizon River Flow Forecasting Using Machine Learning

Mathieu Schneider

30.12.2025

Abstract

This study investigates the feasibility and performance of machine learning methods for multi-horizon river flow forecasting using hourly hydrological and meteorological data. A unified forecasting framework is developed to predict river flow up to 72 hours ahead based on rolling historical windows and engineered long-term summary features. Four model families are evaluated under identical conditions: Random Forests, XGBoost, Long Short-Term Memory (LSTM) networks, and Gated Recurrent Units (GRU). Models are compared across two input configurations, one relying solely on weather information and another additionally incorporating data from other river stations.

Results show that all models exhibit predictive power beyond a naïve benchmark, with forecasting accuracy highest at short horizons and degrading systematically as the horizon increases. XGBoost achieves the best overall performance, closely followed by LSTM, while GRU and Random Forests perform less well. Incorporating additional river stations does not improve accuracy and often slightly degrades performance, suggesting that unfiltered spatial information may introduce noise. Overall, the findings highlight the importance of model expressiveness, careful input selection, and horizon-specific evaluation in short-term hydrological forecasting, while providing a reproducible framework for future extensions.

1 Introduction

Accurate forecasting of river flow plays a crucial role in water resource management, flood risk mitigation, hydropower planning, and environmental protection. Being able to anticipate how river flows will evolve over the coming hours or days allows authorities and operators to take proactive measures rather than responding after critical thresholds are reached. However, river flow dynamics result from complex and highly nonlinear interactions between meteorological conditions, hydrological processes, and seasonal cycles, which makes reliable short- to medium-term forecasting particularly challenging.

The purpose of this project is to examine whether machine learning models can effectively forecast river flow several hours ahead and to analyze how predictive accuracy evolves as the forecast horizon increases.

The central research question addressed in this study is therefore the following:

> “Can machine learning models reliably forecast river flow up to 72 hours ahead using historical water and weather station data, and how does their predictive performance degrade across forecast horizons?”

More specifically, this project aims to investigate:

- Whether different model families—namely tree-based models and recurrent neural networks—exhibit systematically distinct forecasting behaviors.
- Whether incorporating information from additional river stations improves forecast accuracy or instead introduces noise.

- How forecast errors evolve as the prediction horizon increases, and whether certain models display greater temporal stability than others.

To address these questions, the project pursues four main objectives. First, it develops a unified forecasting framework capable of producing multi-horizon river flow forecasts up to 72 hours ahead using hourly data. Second, it compares several machine learning model families—including Random Forests, XGBoost, LSTM, and GRU—under identical data and evaluation conditions. Third, it evaluates forecast accuracy and stability using both global performance metrics and horizon-specific error measures. Finally, it assesses the informational value of additional inputs by contrasting a baseline configuration relying solely on weather data with an extended configuration that also incorporates data from other river stations.

Outline of the Report

The remainder of this report is organized as follows. Section 2 presents the data sources, preprocessing procedures, and feature construction. Section 3 describes the methodological framework, including model architectures, forecasting setup, hyperparameter optimization, and evaluation strategy. Section 4 reports the empirical results, combining descriptive performance comparisons with statistical analyses. Section 5 discusses the main findings, outlines the limitations of the study, proposes directions for future research, and concludes.

2 Data Treatment

The dataset used in this study is composed of two main sources: water station data and weather station data. Water stations are primarily distributed across the territory of the canton of Vaud and provide measurements of river flow. Weather stations are also mainly located within the canton of Vaud and record precipitation levels expressed in millimeters.

Data sources

Water station data are obtained from open-access datasets available on the official website of the Vaud [1] canton and were downloaded in CSV format. All stations listed on the platform were initially considered, except for those directly managed by the federal government. These stations were excluded due to their limited number and differences in data structure, which would have required additional harmonization. After this filtering step, approximately 30 water stations were retained in the raw dataset.

Weather station data were collected from the corresponding official platform [2]. Specifically, data from the "automatic stations" network were used. Stations located in the French-speaking region of Switzerland were selected to ensure sufficient spatial coverage of the study area. In total, precipitation data from 27 weather stations were collected.

The initial objective was to construct a dataset spanning the period from 2000 to 2025. However, as discussed later, data availability constraints and the presence of missing values substantially reduce the effective temporal coverage.

Variables and feature construction

All station measurements are recorded at an hourly frequency using Unix timestamps. For water stations, the raw variable corresponds to hourly river flow measured in cubic meters per hour. For weather stations, the raw variable corresponds to hourly precipitation measured in millimeters.

In addition to these raw variables, several derived features were constructed to enrich the information available to the models. For each weather station, two rolling variables were added at the hourly level: the cumulative precipitation over the previous 14 days and over the previous 30 days. Similarly, for each water station, two rolling averages were computed: the mean river flow over the past 14 days and over the past 30 days. The purpose of these aggregated variables is to facilitate the capture of longer-term trends and delayed hydrological responses that may not be fully represented within short historical windows.

Seasonal effects were also explicitly modeled. For each observation, the week number within the year was computed. To encode this seasonal information in a continuous and cyclic manner, the sine and cosine transformations of the week number were added as additional variables. This representation allows models to identify the position within the annual cycle without introducing artificial discontinuities between the end and the beginning of the year.

Data availability, station selection, and missing values

We imported data from 2000 to 2025 but a lot of data were missing. Either due to bugs, maintenance or that the measurement stations was installed later than 2000.

The first preprocessing step consisted in identifying, for each water station, the longest time interval over which flow data are continuously available. Once this reference interval was determined, a subset of other water stations and weather stations exhibiting continuous data over the same period was selected. These stations were then used as external explanatory variables for forecasting the target water station. As a result, each water station is associated with its own available time span and a corresponding set of auxiliary stations.

This procedure revealed two main issues. First, several stations exhibited an excessive proportion of missing values and were therefore excluded from further analysis. After this filtering step, 20 water stations remained usable. Second, even within the longest continuous intervals, small gaps of missing values were still present, which limited the usable length of the dataset.

To mitigate this issue, a targeted imputation strategy was implemented. Short sequences of missing values were filled using linear interpolation based on the mean of the last observed value before the gap and the first observed value after it. This approach was applied exclusively to short gaps: missing sequences were filled only when their duration did not exceed 24 consecutive hours. Given the overall size of the dataset, imputing gaps of at most one day was deemed unlikely to materially distort the statistical properties of the data while allowing a substantial increase in the number of usable observations.

These preprocessing steps were applied consistently across all water stations. The outcome is, for each station, the longest feasible continuous dataset accompanied by a set of water and weather stations sharing the same availability window and suitable for model input.

Final dataset selection

In this study, the empirical analysis focuses on a single water station. For this station, the final dataset consists of approximately 1,600 days of continuous observations, corresponding to roughly 38,400 hourly measurements. Over this period, data from three weather stations and six additional water stations are available without interruption and are therefore included as explanatory inputs in the forecasting models.

3 Methodology

The goal in this paper is to forecast the debit level of rivers multiple hours in the future and then to evaluate the accuracy and efficiency of these forecasts. We concentrate ourself, to forecast up to 72 hours after the last observed data. This forecast horizon is common for weather models, this is why we choose it as default. Other forecast horizons could have been used but due to a lack of computation power, no time has been found yet to do so.

We face multiple challenges in this paper, first is data heterogeneity. We have multiple input types in our data set (water debit from rivers, weather precipitation, seasonal information). The relationship between these data and the water debit in the river is not obvious and might not be linear.

The second challenge we face is time dependency. Information collected 3 hours before the forecast is probably relevant but the long-term influence of past information is also unclear and different from input to input.

Lastly, we forecast multiple horizon ahead of the last observed value. Consistency of the forecast robustness is searched and key.

To address these challenges, we will use and compare 4 different models: LSTM, GRU, random forests and XGBoost forest. Then we will try to fine tune their hyperparameters and compare their accuracy and stability across horizon to select the best model architecture.

3.1 Approach

General framework

This study adopts a supervised learning framework in which models are trained on historical observations to forecast river flow multiple hours into the future. For each model, the output consists of a 72-hour-ahead forecast of river flow following the last observed data point. Inputs include past flow values of the target station, past observations from selected weather and water stations, as well as seasonal information. Seasonal effects are encoded through the sine and cosine transformations of the week number, allowing models to identify the position within the annual cycle in a continuous manner.

Inputs are structured using rolling windows of 72 hours. In other words, each model uses the previous 72 hourly observations to predict the subsequent 72 hours. This windowing strategy substantially reduces the dimensionality of the problem, limits computational costs, and may help filter out noise from very distant past observations that contribute little predictive power. To compensate for the restricted temporal window, additional long-term summary variables are included: rolling means over the past 14 and 30 days for water stations, and rolling precipitation sums over the past 14 and 30 days for weather stations. This design allows the models to retain longer-term hydrological and meteorological conditions while keeping the input space manageable.

Tree based models

The first class of models considered consists of Random Forests and XGBoost trees. These tree-based approaches are well suited to heterogeneous and nonlinear data structures and therefore serve as natural baseline and exploratory models in this context. However, they do not possess an inherent mechanism for modeling temporal dependencies. As a result, their ability to capture delayed effects relies heavily on the engineered long-term summary variables (14-day and 30-day aggregates), which provide indirect information beyond the 72-hour input window.

Neural networks

Recurrent neural networks are also employed due to their suitability for time-series forecasting tasks. These models are capable of capturing nonlinear relationships while explicitly modeling temporal dependencies through internal memory mechanisms. Two architectures are tested: Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRU). LSTM models are designed to retain longer-term information but typically require more data and computational resources than GRU models. Comparing these two architectures allows us to assess whether the additional complexity of LSTM networks improves forecast accuracy or instead introduces instability or overfitting.

Configuration

Models are evaluated under different input configurations. Configuration A forecasts river flow using past values of the target station and weather station data only. Configuration B extends this setup by additionally incorporating data from other water stations. Comparing these two configurations allows us to assess whether including hydrological information from neighboring stations improves forecast accuracy or introduces noise. Due to constraints in time and computational resources, the empirical analysis focuses exclusively on these two configurations.

Two additional configurations were initially envisioned (table 1). Configuration C would extend Configuration B by systematically selecting and removing stations to assess the contribution of individual inputs and identify sources of noise. Configuration D would explore alternative window lengths and forecast horizons. In the present study, the lookback window and forecast horizon are both fixed at 72 hours, but varying these parameters could provide further insight into temporal dynamics and model sensitivity.

Evaluation metrics and validation strategy

Model performance is primarily evaluated using the Mean Squared Error (MSE). During parameter optimization, each model aims to minimize the multi-horizon MSE. The dataset is divided into three subsets: a training set (70% of the observations), a validation set (20%), and a testing set (10%).

Models are trained on the training set and evaluated on the validation set across multiple hyperparameter configurations. Performance metrics computed on the validation set include:

- The global (mutlihorizon) MSE.
- The global (mutlihorizon) R2 squared for interpretation.
- horizon-specific MSE and R2.
- The variance across horizons of the MSE.

Evaluating models on the validation set rather than the training set mitigates the risk of overfitting. Together, these metrics provide insight into both overall predictive performance and the degradation of accuracy as the forecast horizon increases.

To further analyze temporal degradation and hyperparameter sensitivity, two ordinary least squares (OLS) regressions are conducted. The first regresses per-horizon MSE on the forecast horizon to quantify how errors increase over time. The second regresses global MSE on tested hyperparameter configurations to assess whether certain parameters exhibit systematic linear effects on performance.

After completing these analyses, the best hyperparameter configuration for each model is selected based on validation-set MSE. Each selected model is then evaluated once on the testing set to obtain an unbiased estimate of out-of-sample performance. Only a single configuration per model is tested at this stage to avoid selection bias and overfitting.

Hyperparameters optimization

Each model is trained multiple times using different hyperparameter configurations to identify performant combinations. The optimization procedure begins with five random configurations sampled from predefined parameter ranges (Table 2–3–4–5). These ranges are deliberately restricted based on preliminary experimentation to limit computational costs.

Following the random search phase, hyperparameter tuning proceeds via Bayesian optimization using an Expected Improvement (EI) acquisition function. This approach balances exploitation of regions with strong performance and exploration of regions with high uncertainty.

After the initial random evaluations, the optimizer progressively concentrates on promising areas of the parameter space, leading to a more efficient use of computational resources than pure random search.

For Random Forests and XGBoost models, the optimization consists of five random evaluations followed by 45 Bayesian optimization iterations. For LSTM and GRU models, five random evaluations are followed by 30 Bayesian iterations to reduce computational burden. For each evaluated configuration, performance is measured on both the training and validation sets, and the final model selection is based on minimizing validation-set MSE.

3.2 Implementation

The methodological steps described above are implemented using a modular Python codebase that covers data processing, model training, evaluation, and statistical analysis. The implementation relies on several open-source libraries, including NumPy and pandas for data manipulation, scikit-learn and statsmodels for statistical modeling and evaluation, TensorFlow/Keras for deep learning architectures (LSTM and GRU), XGBoost for gradient-boosted trees, and scikit-optimize for Bayesian hyperparameter optimization. Additional system utilities such as os, inspect, and uuid are used for experiment tracking and file management.

The execution follows a structured pipeline. After data collection and cleaning, the dataset is transformed into windowed input-output pairs. These windowed datasets are then split into training, validation, and testing subsets. For a given hyperparameter configuration, models are trained on the training set and evaluated on the validation set. Hyperparameters are refined through successive random and Bayesian optimization loops. Once all iterations for a given model type are completed, the best-performing configuration is selected.

For tree-based models, a separate model is trained for each forecast horizon at every iteration. In contrast, neural network architectures produce a single model with a multi-output structure, generating forecasts for all horizons simultaneously.

In summary, the analysis strictly separates training, validation, and testing stages. Model selection is based exclusively on validation performance, thereby reducing the risk of overfitting. Testing data are used only once to assess final performance, while all auxiliary analyses, including OLS regressions, rely solely on validation-set outcomes. This design ensures a robust and interpretable evaluation of forecasting accuracy.

4 Results

4.1 OLS results

Linear relationships between hyperparameters and performance

This section first examines the ordinary least squares (OLS) regressions linking hyperparameters to model performance, before analyzing the effect of increasing forecast horizons on prediction error. Finally, the best-performing specifications for each model family are presented and compared in terms of accuracy and temporal stability.

For tree-based models, namely Random Forests and XGBoost, the majority of hyperparameters do not exhibit statistically significant linear relationships with global MSE. This does not imply that these parameters are unimportant, but rather that their influence is likely nonlinear and interdependent. Simply increasing the number of trees does not systematically improve predictive performance.

One notable exception concerns model complexity. In Configuration A, the maximum tree depth shows a positive and statistically significant association with MSE for both Random Forests and XGBoost, with coefficients of 0.04 and 0.032 respectively ($p < 0.05$ in both cases)(Table 6–7). This indicates that increasing model complexity tends to degrade performance, consistent with overfitting behavior.

When additional water station data are included (Configuration B), the same pattern persists: tree depth remains statistically significant ($p < 0.05$) (Table 10–11), indicating continued exposure to overfitting. However, the magnitude of the coefficients decreases substantially, to 0.0115 for Random Forests and 0.0078 for XGBoost. This suggests that increasing the size of the input dataset partially mitigates the negative impact of model complexity.

For recurrent neural networks, forecasting performance is primarily driven by optimization and regularization choices rather than architectural capacity. Across both LSTM and GRU models, the learning rate exhibits a consistently negative and statistically significant relationship with global MSE in all configurations (Table 8–9). In Configuration A, the coefficient on log (learning rate) equals -0.0191 for LSTM and -0.0230 for GRU ($p < 0.01$ in both cases). In contrast, architectural parameters such as the number of units and layers are not statistically significant ($p > 0.3$), indicating that increasing network size does not systematically improve performance.

Differences emerge in training dynamics between the two architectures. In the lower-information setting (Configuration A), GRU models benefit from longer training and weaker regularization. Significant negative coefficients are observed for log(epochs) (-0.0108, $p = 0.018$), log(patience) (-0.0229, $p = 0.006$), and dropout (-0.0859, $p = 0.005$). By contrast, LSTM performance does not significantly depend on training duration (log(epochs): -0.0022, $p = 0.707$), suggesting that LSTM models reach their optimal generalization point earlier and are more prone to overfitting.

When additional river station data are included (Configuration B) (Table 12–13), hyperparameter sensitivity decreases for both recurrent architectures. Learning rate remains statistically significant ($p < 0.01$), while most training-related parameters lose significance. This indicates a stabilization of the learning process in higher-information settings. Overall, these results suggest that GRU models exhibit a slower onset of overfitting than LSTM models and therefore benefit more from softer training constraints when information is limited.

MSE degradation and horizon stability

To assess how forecast accuracy evolves over time, an OLS regression of per-horizon MSE on the forecast horizon is estimated. As expected, a strong and statistically significant relationship is observed between forecast horizon length and prediction error (Table 14).

In Configuration A, Random Forests exhibit the smallest increase in MSE across horizons, with a coefficient of 0.0014. This is followed by GRU (0.0043), LSTM (0.0049), and XGBoost (0.005). This does not imply that Random Forests are more accurate at longer horizons; rather, their performance, whether good or poor, degrades more slowly as the forecast horizon increases.

When additional water station data are included (Configuration B) (Table 15), the pattern changes. While all coefficients remain statistically significant, their magnitudes increase for GRU (0.004), XGBoost (0.006), and LSTM (0.0066), whereas the Random Forest coefficient remains largely unchanged. Including additional stations therefore increases horizon sensitivity for XGBoost and LSTM models, with MSE rising more steeply over time. While definitive conclusions cannot be drawn at this stage, the analysis of the best-performing models in the next section provides further insight.

4.2 Analyze of the best models

This section presents the performance of the best specification identified for each model family. Model specifications are reported in table 16–17. Based on global MSE, models can be ranked from weakest to strongest as follows: Random Forest, GRU, LSTM, and XGBoost (figure 1). The same ranking applies when comparing global R2 values (figure 2).

Overall, with R2 values ranging between 0.3 and 0.35, the models exhibit non-negligible predictive power and could be used as forecasting tools, albeit with moderate accuracy. A notable and somewhat unexpected result is that both global MSE and worsen in Configuration B. The inclusion of additional water stations does not appear to add sufficient explanatory power to compensate for the noise introduced.

One possible explanation is that the flow of the additional water stations is weakly correlated with the target station. Although the stations are geographically close and belong to nearby watersheds, they are located on distinct river systems without direct hydrological connections. However, this result does not support definitive conclusions. The comparison here contrasts the use of all available water stations with none; it is possible that some stations contribute useful information while others add noise. Moreover, this finding should not be generalized to other forecasting contexts, where hydrological relationships may differ.

Another noteworthy result is that Random Forest performance remains largely unaffected by the inclusion of additional data. This may indicate that the trees effectively ignore non-informative inputs when they do not improve MSE, thereby maintaining stable performance.

Examining MSE and R2 across forecast horizons provides a clearer picture of temporal dynamics (figure 3–5). Prediction accuracy deteriorates rapidly as the horizon increases, before stabilizing after approximately 30 hours. Differences across models are most pronounced at short horizons. XGBoost consistently outperforms other models, while LSTM follows closely behind, exhibiting strong early-horizon accuracy and slightly better performance at longer horizons. These results also clarify the earlier findings regarding Random Forests: although their error grows slowly over time, their overall accuracy remains substantially lower than that of more expressive models.

Finally, the impact of Configuration B is again evident (figure 4–6). Across most horizons, MSE and R2 are either unchanged or slightly worse when additional water stations are included. The negative effect is particularly visible for LSTM models at the longest forecast horizons, reinforcing the conclusion that unfiltered input expansion can degrade long-term forecasting performance.

5 Discussion and Conclusion

This study demonstrates that machine learning models are able to capture a non-negligible share of river flow dynamics at short forecasting horizons. All tested approaches exhibit predictive power beyond a naïve benchmark, confirming the feasibility of multi-horizon flow forecasting in this setting. While overall performance remains moderate, results are consistent across evaluation metrics, lending credibility to the empirical findings.

Clear differences emerge across model families. Random Forests provide stable but comparatively weaker performance, whereas recurrent neural networks improve short-term accuracy. Among all approaches, XGBoost achieves the best overall results, closely followed by the LSTM, while GRU models occupy an intermediate position.

Comparing configurations, the inclusion of additional water stations (Config B) does not improve performance relative to using weather information alone (Config A). On the contrary, results slightly deteriorate, suggesting that unfiltered spatial hydrological information may introduce noise rather than useful signal for the target station. This outcome appears station-specific and should not be generalized without further empirical validation.

Finally, forecasting accuracy degrades systematically as the forecast horizon increases. This degradation is steep at short horizons and stabilizes beyond approximately one day ahead, indicating that informative temporal signals are rapidly exhausted. Differences across models are most pronounced at early horizons, where more expressive architectures exhibit clear advantages.

5.1 Limitations of the Study

Several limitations should be considered when interpreting these results. First, computational constraints strongly shaped the experimental design. Due to limited computing resources, the analysis focuses on a single target water station and restricts attention to two input configurations. Hyperparameter search spaces and the depth of Bayesian optimization were deliberately constrained to keep training times manageable, potentially preventing some models from reaching their full performance. As shown in the best model specifications, several hyperparameters lie at the boundaries of the predefined search ranges suggesting that improved configurations may exist beyond the explored space (table 2–3–4–5, 16–17).

Second, the generalizability of the results remains uncertain. Conclusions are drawn from a single hydrological context, and both absolute performance levels and the relative ranking of model families may differ across river systems with distinct climatic, hydrological, or spatial characteristics.

Third, input relevance was not explicitly assessed. Although multiple weather and water station variables were included, no formal feature-importance or variable-contribution analysis was conducted. Consequently, informative inputs may be diluted by less relevant variables, particularly in configurations with high-dimensional input spaces.

Fourth, the temporal design is fixed. Both the lookback window and the forecast horizon are set to 72 hours, and no sensitivity analysis was performed to evaluate alternative temporal configurations. Different window lengths or forecast spans could lead to different conclusions regarding model suitability and stability.

Finally, data availability and quality impose additional constraints. Historical coverage is limited, requiring partial imputation of short missing sequences, and the spatial coverage of both weather and water stations is incomplete. Although the imputation strategy is conservative, these factors reduce the effective information content of the dataset and may limit achievable forecasting accuracy.

5.2 Directions for Future Work

Several extensions could naturally build on this study and address its main limitations. A first priority is scaling the analysis across multiple water stations. Applying the same framework to a broader set of locations would allow comparison across hydrological contexts and provide insight into the robustness of model rankings under varying watershed characteristics.

A second avenue concerns input relevance and interpretability. For tree-based models, feature-importance techniques such as permutation importance or SHAP values could identify the most informative variables. For neural networks, sensitivity or ablation analyses would help quantify the contribution of individual inputs or groups of stations. This line of work would directly support station selection strategies, as originally envisioned in Configuration C (table 1).

Further research should also explore alternative temporal designs. Varying the lookback window length and forecast horizon would clarify how much historical information is truly required and how predictive skill evolves across shorter or longer forecast spans.

In addition, hyperparameter exploration could be expanded. Relaxing parameter bounds and extending Bayesian optimization phases would enable a more thorough investigation of model capacity and its interaction with overfitting, potentially leading to different conclusions regarding optimal architectures.

Finally, richer data integration represents a promising direction. Incorporating additional meteorological variables—such as temperature, snow indicators, or soil moisture—could substantially enhance predictive performance. Access to longer and cleaner historical datasets would further strengthen the empirical foundation of future analyses.

5.3 Final Conclusion

This study shows that machine learning methods can be effectively applied to multi-horizon river flow forecasting, yielding meaningful predictive performance, particularly at short forecast horizons. While overall accuracy remains moderate, the results confirm that data-driven models can extract useful temporal and meteorological signals from heterogeneous hydrological datasets.

Clear performance differences emerge across model families. More flexible models, especially gradient-boosted trees and long short-term memory neural network, outperform simpler tree-based approaches, highlighting the importance of model expressiveness when dealing with nonlinear dynamics and temporal dependencies. At the same time, forecasting skill declines rapidly as the horizon increases, emphasizing the intrinsic difficulty of long-term flow prediction and the dominance of short-term information.

A key finding is that adding more information does not automatically improve forecasts. The inclusion of additional water stations without prior selection can introduce noise and degrade performance, underlining the need for careful input design rather than indiscriminate data expansion.

Beyond empirical results, this work delivers a modular and reproducible forecasting framework that combines systematic model evaluation, multi-horizon metrics, and rigorous validation procedures. As such, it provides a solid foundation for future research aimed at improving hydrological forecasting through better data integration, model design, and interpretability.

Data source

1. Canton de Vaud. *Hydrological Data Platform.* Available at: `https://vhv.ch/xt_vh_718536/index.php?ck3_page_reloaded=1`.

2. MétéoSuisse. *Measured Values and Measurement Networks.* Available at: `https://www.meteosuisse.admin.ch/services-et-publications/applications/valeurs-mesurees-et-reseaux-d`
`html#param=messwerte-niederschlag-10min&station=NEU&table=false&chart=hour&compare=n`.

A Appendix: Tables and Figures

This appendix reports tables referenced in the main text.

A.1 Configuration table

Table 1: Input configurations and associated research objectives

Config	**Input Data**	**Key Characteristics**	**Purpose / Research Question**
A – Weather Only	Precipitation from relevant weather stations, seasonal features, and long-term precipitation aggregates	No information from other river stations; relies only on meteorological forcing and recent river history	Establish a **baseline**: how well river flow can be predicted from weather information alone
B – Weather + Other Rivers	Weather inputs (as in A) plus flow data from other water stations (short-term and 14/30-day means)	Introduces spatial hydrological dependencies and upstream/parallel river information	Test whether **river network information** improves flow forecasts, especially for peak events
C – Weather Station Selection	Same structure as A or B, but with different subsets of weather stations (e.g. closest only vs. all stations)	Evaluates sensitivity to spatial selection of meteorological inputs	Assess the **importance of weather station choice** and spatial representativeness
D – Lookback Window Variation	Same inputs as A or B, but with different historical window lengths (e.g. 24h, 48h, 72h) and forecast horizons	Changes how much recent history the model observes	Analyze the **impact of memory length** on forecasting accuracy and temporal dynamics

A.2 Hyperparameters range and definition

Table 2: Random Forest (Tree) hyperparameter search space

Hyperparameter	**Search Range**	**Role**
Number of trees	5 – 150	Controls ensemble size and variance reduction
Maximum depth	1 – 10	Controls model complexity
Minimum samples per leaf	1 – 10	Prevents overfitting by enforcing minimum leaf size

Table 3: XGBoost hyperparameter search space

Hyperparameter	**Search Range**	**Role**
Number of trees	1 – 150	Number of boosting rounds
Maximum depth	1 – 6	Controls tree complexity
Learning rate	0.05 – 0.2	Controls contribution of each tree
Subsample ratio	0.7 – 1.0	Fraction of samples used per tree
Column subsample ratio	0.7 – 1.0	Fraction of features used per tree

Table 4: GRU hyperparameter search space

Hyperparameter	**Search Range**	**Role**
Number of units	16 – 96	Controls model capacity
Number of layers	1 – 3	Controls temporal depth
Dropout	0.0 – 0.4	Regularization
Recurrent dropout	0.0 – 0.3	Regularization of gating mechanism
Batch size	32 – 256	Training efficiency
Number of epochs	3 – 40	Maximum training iterations
Early stopping patience	2 – 8	Controls generalization via early stopping
Learning rate	10^{-4} – 3×10^{-3}	Optimizer step size

Table 5: LSTM hyperparameter search space

Hyperparameter	**Search Range**	**Role**
Number of units	16 – 96	Controls memory capacity
Number of layers	1 – 3	Controls network depth
Dropout	0.0 – 0.4	Regularization between layers
Recurrent dropout	0.0 – 0.3	Regularization of recurrent connections
Batch size	32 – 256	Training stability and efficiency
Number of epochs	3 – 40	Maximum training iterations
Early stopping patience	2 – 8	Stops training when validation stalls
Learning rate	10^{-4} – 3×10^{-3}	Optimizer step size

A.3 OLS on Hyperparameters Config A

Table 6: Hyperparameter OLS on Global MSE (Config A, Random Forest)

Variable	**Coefficient**	**Std. Error**	**z-stat**	**p-value**
log(number of trees)	−0.0015	0.003	−0.519	0.604
maximum depth	0.0405	0.002	26.540	0.000
log(min samples per leaf)	−0.0022	0.005	−0.457	0.647
constant	0.0707	0.012	5.765	0.000

Table 7: Hyperparameter OLS on Global MSE (Config A, XGBoost)

Variable	**Coefficient**	**Std. Error**	**z-stat**	**p-value**
log(number of trees)	0.0075	0.004	1.765	0.078
maximum depth	0.0327	0.004	7.853	0.000
log(learning rate)	0.0070	0.010	0.734	0.463
subsample ratio	0.0655	0.057	1.154	0.248
column subsample ratio	−0.1365	0.058	−2.343	0.019
constant	0.1190	0.071	1.686	0.092

Table 8: Hyperparameter OLS on Global MSE (Config A, GRU)

Variable	**Coefficient**	**Std. Error**	**z-stat**	**p-value**
log(learning rate)	−0.0230	0.004	−6.528	0.000
log(batch size)	−0.0055	0.006	−0.965	0.335
log(epochs)	−0.0108	0.005	−2.360	0.018
log(patience)	−0.0229	0.008	−2.746	0.006
log(units)	0.0051	0.005	0.974	0.330
number of layers	−0.0013	0.005	−0.271	0.786
dropout	−0.0859	0.031	−2.802	0.005
recurrent dropout	0.0170	0.029	0.587	0.557
constant	0.0502	0.045	1.107	0.268

Table 9: Hyperparameter OLS on Global MSE (Config A, LSTM)

Variable	**Coefficient**	**Std. Error**	**z-stat**	**p-value**
log(learning rate)	−0.0191	0.005	−4.222	0.000
log(batch size)	0.0168	0.008	2.148	0.032
log(epochs)	−0.0022	0.006	−0.375	0.707
log(patience)	−0.0274	0.011	−2.566	0.010
log(units)	−0.0049	0.006	−0.816	0.414
number of layers	0.0053	0.007	0.818	0.413
dropout	−0.1415	0.061	−2.312	0.021
recurrent dropout	−0.0219	0.036	−0.613	0.540
constant	0.0032	0.056	0.057	0.954

A.4 OLS on Hyperparameters Config B

Table 10: Hyperparameter OLS on Global MSE (Config B, Random Forest)

Variable	Coefficient	Std. Error	z-stat	p-value
log(number of trees)	−0.0017	0.001	−1.229	0.219
maximum depth	0.0115	0.001	16.252	0.000
log(min samples per leaf)	0.0004	0.001	0.262	0.793
constant	0.0951	0.006	15.173	0.000

Table 11: Hyperparameter OLS on Global MSE (Config B, XGBoost)

Variable	Coefficient	Std. Error	z-stat	p-value
log(number of trees)	−0.0035	0.003	−1.089	0.276
maximum depth	0.0078	0.002	5.148	0.000
log(learning rate)	0.0085	0.005	1.772	0.076
subsample ratio	−0.0034	0.021	−0.159	0.874
column subsample ratio	−0.0041	0.028	−0.147	0.884
constant	0.1137	0.023	4.861	0.000

Table 12: Hyperparameter OLS on Global MSE (Config B, GRU)

Variable	Coefficient	Std. Error	z-stat	p-value
log(learning rate)	−0.0162	0.003	−5.165	0.000
log(batch size)	0.0048	0.003	1.364	0.173
log(epochs)	−0.0038	0.002	−1.536	0.125
log(patience)	−0.0053	0.006	−0.848	0.397
log(units)	−0.0048	0.005	−1.045	0.296
number of layers	3.77×10^{-5}	0.003	0.011	0.991
dropout	−0.0694	0.026	−2.699	0.007
recurrent dropout	0.0076	0.026	0.296	0.767
constant	0.0161	0.035	0.467	0.641

Table 13: Hyperparameter OLS on Global MSE (Config B, LSTM)

Variable	Coefficient	Std. Error	z-stat	p-value
log(learning rate)	−0.0117	0.004	−2.934	0.003
log(batch size)	−0.0016	0.005	−0.344	0.731
log(epochs)	−0.0058	0.003	−2.288	0.022
log(patience)	−0.0089	0.005	−1.879	0.060
log(units)	−0.0094	0.005	−1.728	0.084
number of layers	−0.0040	0.004	−1.117	0.264
dropout	−0.0044	0.021	−0.213	0.832
recurrent dropout	−0.0087	0.019	−0.459	0.646
constant	0.0948	0.054	1.746	0.081

A.5 OLS of horizon on MSE

Table 14: Horizon effect on MSE (Configuration A)

Model	β **(Horizon)**	**Std. Error**	**t-stat**	**p-value**	$\mathbf{R^2}$	**N**
GRU	0.00427	0.00030	14.37	$< 10^{-45}$	0.797	72
LSTM	0.00489	0.00035	14.09	$< 10^{-44}$	0.789	72
Random Forest	0.00148	0.00011	13.36	$< 10^{-40}$	0.782	72
XGBoost	0.00532	0.00049	10.96	$< 10^{-27}$	0.719	72

Table 15: Horizon effect on MSE (Configuration B)

Model	β **(Horizon)**	**Std. Error**	**t-stat**	**p-value**	$\mathbf{R^2}$	**N**
GRU	0.00396	0.00028	14.12	$< 10^{-44}$	0.791	72
LSTM	0.00652	0.00039	16.66	$< 10^{-61}$	0.826	72
Random Forest	0.00148	0.00011	13.36	$< 10^{-40}$	0.782	72
XGBoost	0.00607	0.00056	10.81	$< 10^{-26}$	0.704	72

A.6 Best model specification

Table 16: Best model specifications (Configuration A)

Model	**Global MSE**	**Global** R^2	**Var(MSE)**	**Key Hyperparameters**
GRU	0.0733	0.397	0.000338	Batch size = 204, Units = 70, Epochs = 30, Learning rate = 0.00188, Dropout = 0.343, Patience = 7
LSTM	0.0727	0.402	0.000320	Batch size = 256, Units = 96, Epochs = 39, Learning rate = 0.003, Dropout = 0.40, Patience = 6
XGBoost	0.0726	0.402	0.000783	Max depth = 1, Trees = 105, Learning rate = 0.05, Subsample = 0.7, Column sample = 1.0
Random Forest	0.0925	0.239	0.000188	Max depth = 1, Trees = 15, Min samples per leaf = 2

Table 17: Best model specifications (Configuration B)

Model	**Global MSE**	**Global R^2**	**Var(MSE)**	**Key Hyperparameters**
GRU	0.0746	0.386	0.000197	Batch size = 203, Units = 16, Epochs = 17, Learning rate = 0.00281, Dropout = 0.259, Patience = 5
LSTM	0.0716	0.411	0.000176	Batch size = 256, Units = 96, Epochs = 40, Learning rate = 0.00197, Dropout = 0.192, Patience = 2
XGBoost	0.0712	0.414	0.000627	Max depth = 1, Trees = 150, Learning rate = 0.0538, Subsample = 0.7, Column sample = 0.85
Random Forest	0.0925	0.239	0.000188	Max depth = 1, Trees = 15, Min samples per leaf = 2

A.7 Figures best models

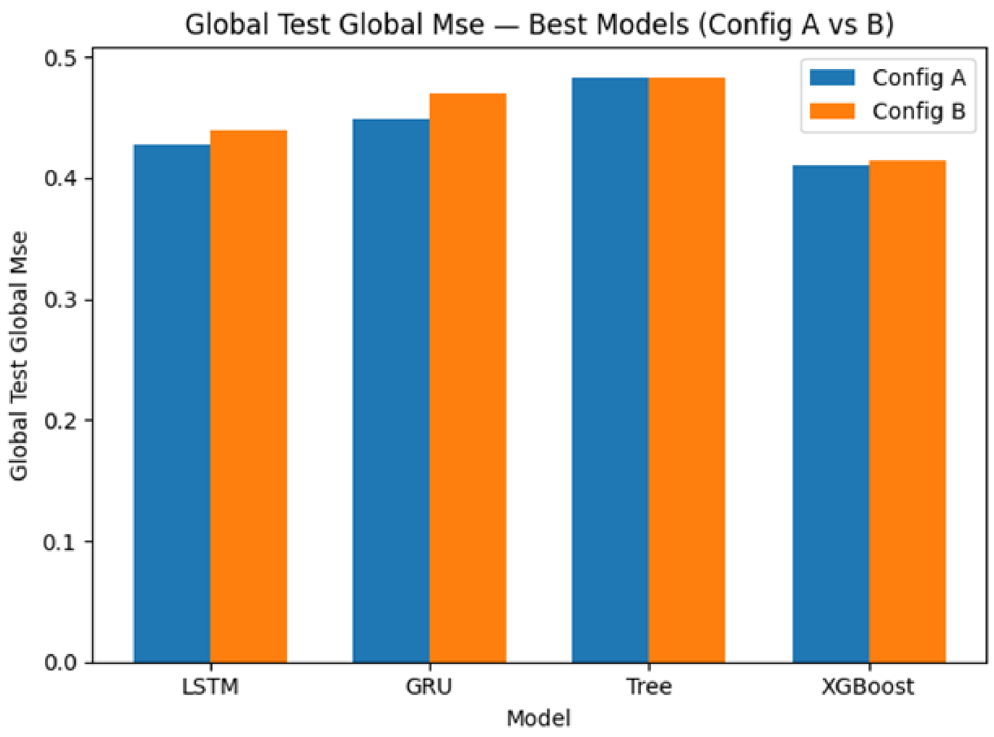

Figure 1: Global MSE for best models.

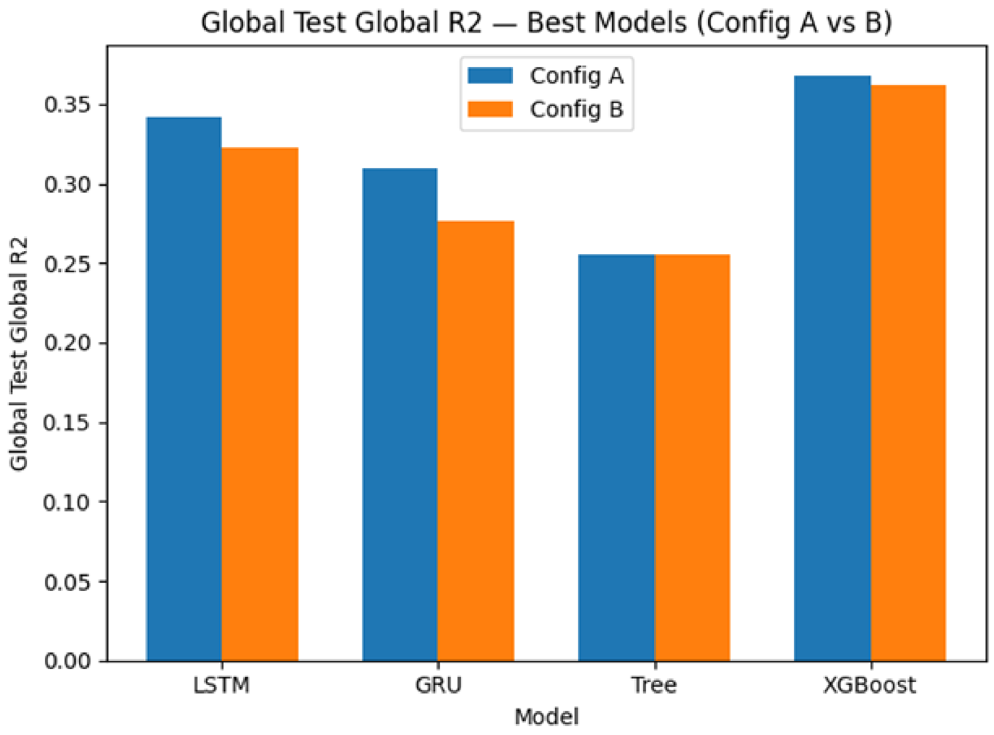

Figure 2: Global R2 for best models.

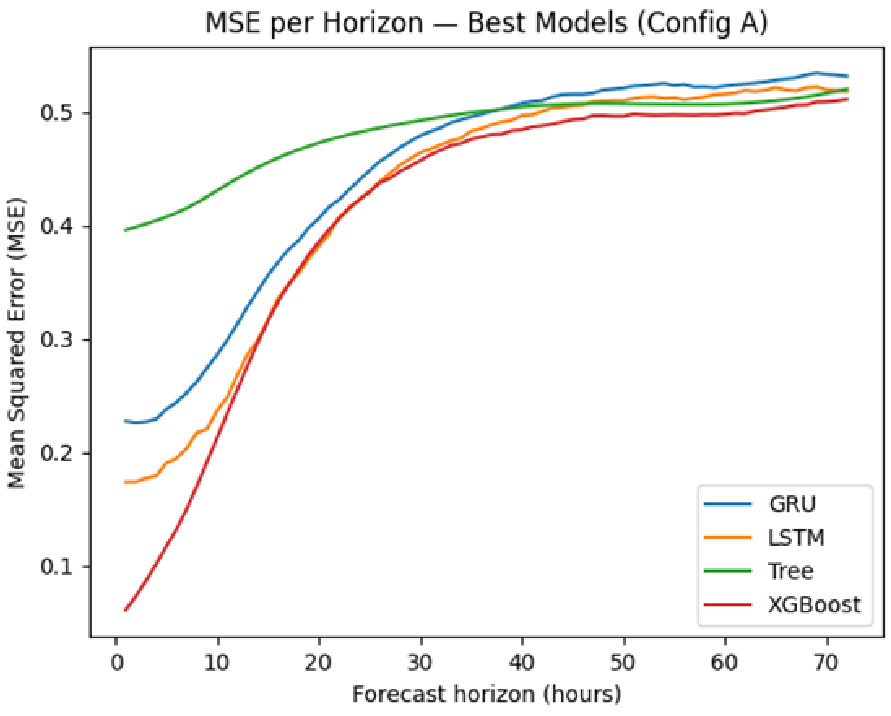

Figure 3: MSE per forecast horizon for best models (Configuration A).

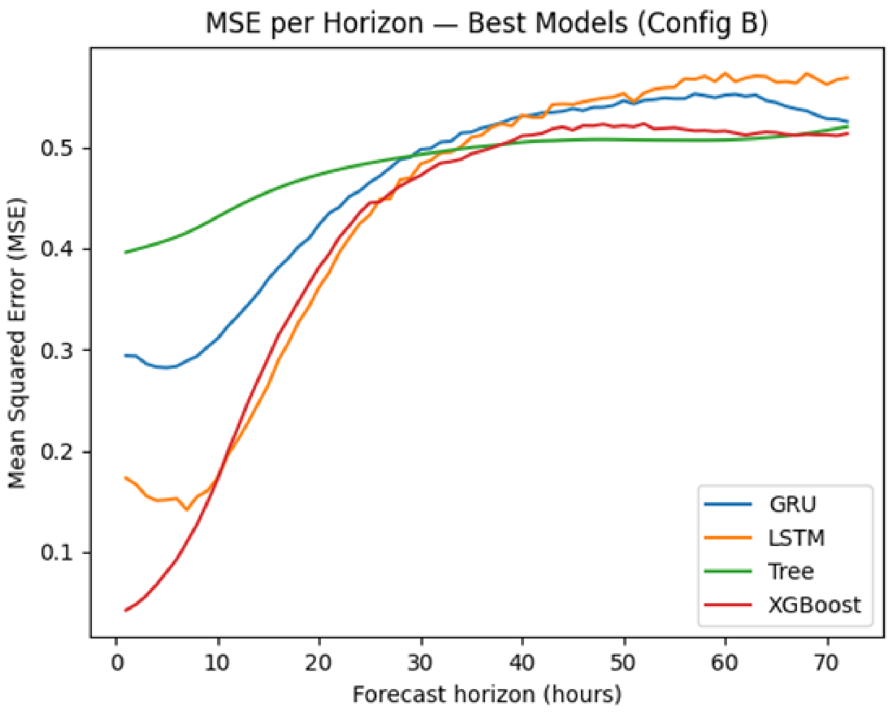

Figure 4: MSE per forecast horizon for best models (Configuration B).

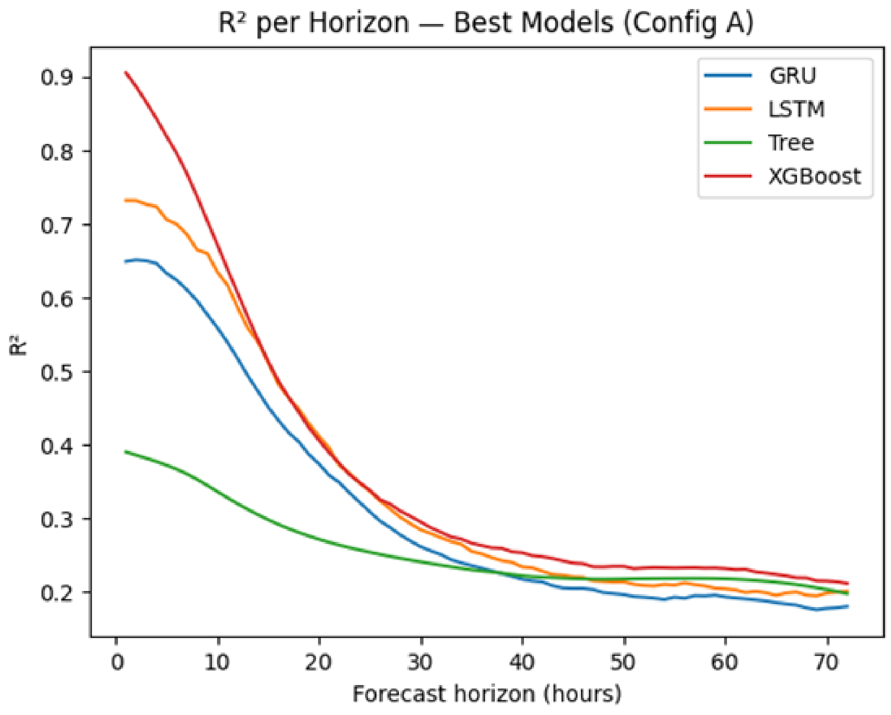

Figure 5: R2 per forecast horizon for best models (Configuration A).

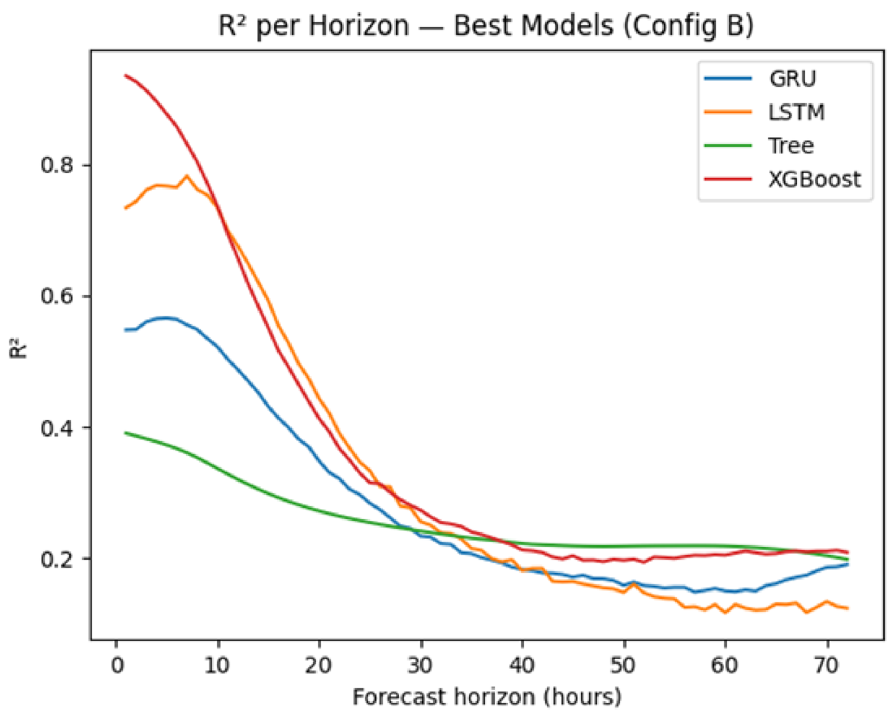

Figure 6: R2 per forecast horizon for best models (Configuration B).

B Appendix: Supplementary tables

B.1 OLS of Hyperparameters on MSE variance Config A

Table 18: Hyperparameter OLS on variance of MSE across horizons (Config A, Random Forest)

Variable	Coefficient	Std. Error	z-stat	p-value
log(number of trees)	−0.0004	0.001	−0.445	0.656
maximum depth	0.0081	0.000	18.829	0.000
log(min samples per leaf)	0.0002	0.002	0.087	0.930
constant	−0.0041	0.004	−1.030	0.303

Table 19: Hyperparameter OLS on variance of MSE across horizons (Config A, XGBoost)

Variable	Coefficient	Std. Error	z-stat	p-value
log(number of trees)	0.0019	0.000	4.974	0.000
maximum depth	0.0030	0.000	7.268	0.000
log(learning rate)	0.0015	0.001	1.688	0.091
subsample ratio	0.0056	0.005	1.185	0.236
column subsample ratio	−0.0116	0.005	−2.328	0.020
constant	0.0007	0.006	0.116	0.908

Table 20: Hyperparameter OLS on variance of MSE across horizons (Config A, GRU)

Variable	**Coefficient**	**Std. Error**	**z-stat**	**p-value**
log(learning rate)	-0.0007	0.000	-1.546	0.122
log(batch size)	-7.36×10^{-5}	0.000	-0.347	0.729
log(epochs)	-0.0005	0.000	-1.456	0.146
log(patience)	0.0004	0.000	1.077	0.281
log(units)	-0.0001	0.000	-0.420	0.674
number of layers	-1.07×10^{-5}	0.000	-0.061	0.951
dropout	0.0008	0.002	0.395	0.693
recurrent dropout	-0.0011	0.002	-0.688	0.492
constant	-0.0020	0.002	-1.241	0.215

Table 21: Hyperparameter OLS on variance of MSE across horizons (Config A, LSTM)

Variable	**Coefficient**	**Std. Error**	**z-stat**	**p-value**
log(learning rate)	-4.79×10^{-5}	4.63×10^{-5}	-1.034	0.301
log(batch size)	3.52×10^{-5}	3.04×10^{-5}	1.158	0.247
log(epochs)	6.85×10^{-5}	2.80×10^{-5}	2.444	0.015
log(patience)	-6.70×10^{-5}	6.53×10^{-5}	-1.027	0.305
log(units)	4.51×10^{-5}	4.45×10^{-5}	1.014	0.311
number of layers	-9.27×10^{-5}	4.18×10^{-5}	-2.218	0.027
dropout	-0.0010	0.000	-3.164	0.002
recurrent dropout	0.0002	0.000	0.708	0.479
constant	1.36×10^{-6}	0.000	0.004	0.997

B.2 OLS of Hyperparameters on MSE variance Config B

Table 22: Hyperparameter OLS on variance of MSE across horizons (Config B, Random Forest)

Variable	**Coefficient**	**Std. Error**	**z-stat**	**p-value**
log(number of trees)	-0.0006	0.000	-1.866	0.062
maximum depth	0.0021	0.000	15.751	0.000
log(min samples per leaf)	0.0002	0.000	0.500	0.617
constant	0.0013	0.001	0.900	0.368

Table 23: Hyperparameter OLS on variance of MSE across horizons (Config B, XGBoost)

Variable	**Coefficient**	**Std. Error**	**z-stat**	**p-value**
log(number of trees)	0.0005	0.000	4.488	0.000
maximum depth	0.0004	4.33×10^{-5}	8.772	0.000
log(learning rate)	0.0005	0.000	2.840	0.005
subsample ratio	0.0002	0.001	0.220	0.826
column subsample ratio	0.0006	0.001	0.609	0.542
constant	-0.0009	0.001	-0.978	0.328

Table 24: Hyperparameter OLS on variance of MSE across horizons (Config B, GRU)

Variable	**Coefficient**	**Std. Error**	**z-stat**	**p-value**
log(learning rate)	-1.17×10^{-5}	4.32×10^{-5}	-0.272	0.786
log(batch size)	-1.39×10^{-5}	2.11×10^{-5}	-0.661	0.509
log(epochs)	2.22×10^{-5}	1.50×10^{-5}	1.483	0.138
log(patience)	6.04×10^{-6}	4.31×10^{-5}	0.140	0.889
log(units)	9.96×10^{-6}	2.47×10^{-5}	0.403	0.687
number of layers	3.51×10^{-5}	3.21×10^{-5}	1.094	0.274
dropout	-0.0004	0.000	-1.546	0.122
recurrent dropout	-0.0003	0.000	-1.306	0.192
constant	9.55×10^{-5}	0.000	0.329	0.742

Table 25: Hyperparameter OLS on variance of MSE across horizons (Config B, LSTM)

Variable	**Coefficient**	**Std. Error**	**z-stat**	**p-value**
log(learning rate)	1.86×10^{-5}	1.42×10^{-5}	1.305	0.192
log(batch size)	-1.53×10^{-5}	1.81×10^{-5}	-0.844	0.399
log(epochs)	-1.25×10^{-5}	9.64×10^{-6}	-1.292	0.196
log(patience)	-3.49×10^{-6}	2.63×10^{-5}	-0.133	0.894
log(units)	-5.87×10^{-5}	3.26×10^{-5}	-1.799	0.072
number of layers	-4.54×10^{-5}	1.20×10^{-5}	-3.767	0.000
dropout	-7.75×10^{-5}	8.51×10^{-5}	-0.911	0.362
recurrent dropout	-4.79×10^{-6}	8.52×10^{-5}	-0.056	0.955
constant	0.0008	0.000	3.157	0.002