Emergent, 2025, X(X), X-X

Link: http://hdl.handle.net/11715/960
Journal Website: http://www.revistas.udb.edu.sv/ojs/
Accepted: Day month year
ISSN 1814-6309

RESEARCH ARTICLE

A study of deep learning models with a low number of parameters for predicting water levels in Vietnam.

Rasoul Islangueriev*, Oumaima El Hemer†, Ngoc Canh Doan‡, Nguyen Le Anh Minh§

*E-mail: rasoul.islangueriev24@gmail.com E-mail: ngoccanh2710@gmail.com
†E-mail: oumaima.elhemer@gmail.com E-mail: anhminh2122000@gmail.com

Abstract
This paper studies the implementation and performance of four deep learning models for predicting water levels in Vietnam. This study focuses on models with a small number of parameters; for our predictions, we use only two: water level and rainfall. As new, increasingly sophisticated deep learning models emerge with increasingly complex architectures, the main difficulty is no longer acquiring high-performance technology, but rather acquiring reliable, accurate data in large quantities to train the models, which is not always possible. Therefore, we will try to determine whether it is possible to have prediction models with few parameters that produce accurate results. Using a small number of parameters also saves computing time, meaning we don't require access to Nvidia graphics cards to train our models. We are using a dataset spanning from 2010 to 2020, a subset of our full dataset. We do this to ensure that our data is stationary. We are comparing four different models: **CNN, Transformer, GNN and a hybrid series model**. We also trained a linear regression model, as we observed a strong linear relationship among the variables. This model served as a benchmark for evaluating the performance of more complex architectures.

Of these four models, the **CNN and the Transformer** are the two best. We use these two models to improve our predictions using two methods: the mean method and the threshold method. Our results show that the **CNN + Transformer model with the mean method** is the most effective. These results pave the way for the use of sober and efficient models in contexts where hydrological data is sparse.
Keywords: CNN, Transformer, GNN, Time Series, Hybrid model

1. Introduction

Flooding is a major cause of damage to infrastructure and agriculture in Vietnam, as well as loss of human life. There are many reasons for this, but the most obvious are its **3,200 km-long coastline**, which exposes the country to bad weather forming in the South China Sea, its tropical climate, which favours storm formation, and the population's distribution in the **Mekong Delta** [16].

Therefore, **Vietnam** is located in a geographical area that makes the country susceptible to flooding, and its population distribution increases collateral damage. Flood prediction plays an important role in enabling decisions to be made to reduce material damage and, if necessary, to relocate people from at-risk areas.

Predicting these phenomena requires access to large quantities of reliable **meteorological data**, which is now possible thanks to the increased availability of

Received: a; **Reviewed:** b; **Accepted:** c

reliable weather-related data and the emergence of deep learning models that can learn and determine the underlying relationships in the data [17].

These models can be built faster and more efficiently than **physical models** that solve systems of complex differential equations requiring modelling of different parameters.

In this paper, we trained and evaluated various deep learning models using time series data on water levels and rainfall collected from seven stations in a region of Vietnam. We also employed methods to enhance the precision of our forecasts.

2. Related works

Time Series Theory

Time series data refers to sequences of observations recorded at successive time intervals. Traditional time series analysis relies heavily on statistical models such as Autoregressive (AR), Moving Average (MA), Autoregressive Integrated Moving Average (ARIMA), and Seasonal ARIMA (SARIMA) models [18].

These models assume stationarity and often require significant preprocessing steps such as differencing and decomposition. Although these approaches offer interpretability and theoretical grounding, their capacity to model nonlinear relationships and high-dimensional multivariate data is limited [19].

In recent years, deep learning approaches have emerged as powerful alternatives. Recurrent neural networks (RNNs) and long-short-term memory (LSTM) networks were among the first to show success in modelling temporal dependencies [20, 21].

More recent architectures such as **Convolutional Neural Networks (CNNs)** [22], **Transformers** [23, 24], and **Graph Neural Networks (GNNs)** [25, 26] have expanded the landscape, allowing the modelling of short and long-range dependencies, as well as spatial-temporal dynamics [10].

These models are particularly advantageous in hydrological and meteorological applications, where time series often involve multiple interdependent variables [1].

Time Series Regression

Time series regression involves predicting a target variable at future time points based on historical values of one or more input features. Classical techniques include linear regression with lagged variables, dynamic regression, and Vector Autoregression (VAR) [27].

These methods typically assume linear relationships and fixed temporal dependencies, which can be limiting in real-world scenarios characterized by noise, nonlinearity, and variable lags. Deep learning-based regression techniques have gained popularity due to their flexibility and precision.

CNNs can capture local trends and patterns in fixed-length windows [22]. Transformers, with their attention mechanisms, are particularly effective in modelling long-term dependencies without the need for recurrence [23].

GNNs are useful when spatial relationships between sensors or stations play a significant role in the evolution of the time series, enabling the fusion of spatial and temporal information into the regression process [2, 26]. The recent literature also highlights the potential of hybrid models that combine different architectures to leverage their complementary strengths [1].

Ensemble techniques, such as **averaging or threshold-based** switching between models, further enhance prediction precision and robustness, particularly in the context of flood forecasting, where both local peaks and overall trends must be captured reliably [28].

3. Theories

Times Series Following Generalized Linear Models

GLM is a generalisation of classical linear regression models. A classical linear regression model establishes a relationship between a target variable Y and a descriptive variable X. This relationship can be generalised to include several descriptive variables, expressed as

$$\mathrm{Y} = \beta_0 + \beta_1 \mathrm{X}_1 + \cdots + \beta_\mathrm{n} \mathrm{X}_\mathrm{n} + \varepsilon$$

where ε is a centred noise term with constant variance. GLMs also use linear modelling, but this modelling does not describe the variable we want to predict directly, rather it describes a transformation of this variable mean through a link function:

$$\mathrm{g}(\mathbb{E}[\mathrm{Y}]) = \beta_0 + \beta_1 \mathrm{X}_1 + \cdots + \beta_\mathrm{n} \mathrm{X}_\mathrm{n}$$

We illustrate the use of GLM on a Titanic dataset and predict the probability of survival according to a person's age, sex and his cabin class.We use for this R, and we downloaded our data set from Kaggle (Titanic dataset from Kaggle)

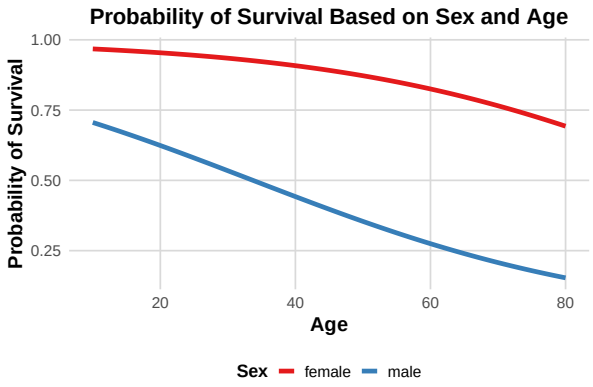


Fig. 1: Probability of survival based on sex and age

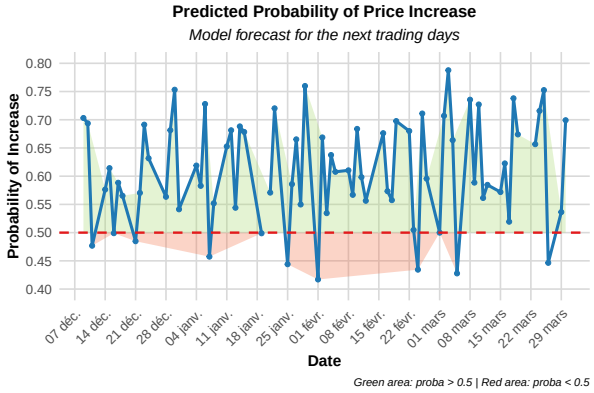


Fig. 2: Predicted Probability of Price Increase

This graph shows that women were more likely to survive the Titanic crash than men.

Regression Models for Binary Time Series

If we want to use time series to predict a binary variable (e.g. win or lose, sick or healthy), we can generate the results seen above. As with GLMs, the variable Y is characterised by parameters $X_1, X_2, \ldots, X_n$ and noise. However, it also depends on its own past. Therefore, we can characterise the expectation of Y's success using the logit function as follows:

$$\begin{aligned}\operatorname{logit}(\mathbb{E}[Y_t]) = \alpha + \sigma_1 Y_{t-1} + \cdots + \sigma_p Y_{t-p} \\ + \beta_1 X_1 + \cdots + \beta_n X_n \quad (1)\end{aligned}$$

We now illustrate the use of a **binary regression model** using a dataset from Kaggle (Microsoft dataset from Kaggle) containing the stock market prices of Microsoft. The objective is to model the **probability that the stock price increases on the following day**, taking into account both historical price movements and market indicators observed on the day t.

Regression Models for Categorical Time Series

Regression models for categorical time series aim to predict a categorical variable observed over time, such as weather conditions (e.g., *Sunny*, *Rainy*, *Cloudy*). These models generalize standard classification approaches by incorporating the temporal dependence of the observations [13]. Let Y_t be a categorical variable representing the state at time t. The conditional probability of Y_t can be modelled using a multinomial logistic regression:

$$\mathbb{P}(Y_t = k \mid X_t, Y_{t-1}, Y_{t-2}, \ldots) = \frac{\exp(\eta_{t,k})}{\sum_{j=1}^{K} \exp(\eta_{t,j})}$$

. To illustrate regression models for categorical time series, we use a weather dataset from Kaggle (weather dataset from Kaggle) containing historical meteorological observations (temperature, humidity, wind speed, etc.). The model is trained and evaluated in `R` using the `nnet::multinom` function.

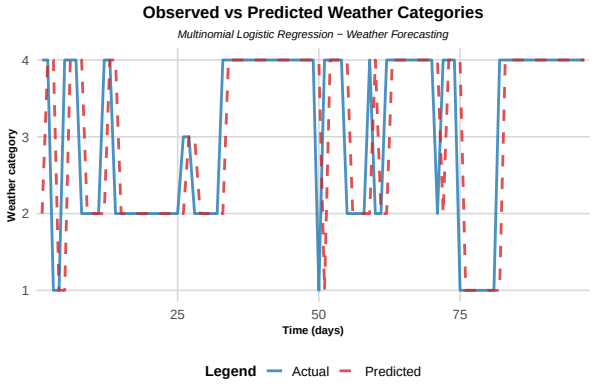


Fig. 3: Observed vs Predicted Weather Categories

Regression Models for Count Time Series

Count time series data refers to the modeling of integer-valued data indexed in time, such as the number of phone calls received per hour or daily hospital admissions. Classical linear regression is not suitable here, as the response variable is a non-negative count [15]. A common model is the Poisson regression, where the conditional mean of the response variable is modelled using a log link function:

$$\mathbb{E}[\mathrm{Y_t} \mid \mathrm{X_t}] = \exp(\beta_0 + \beta_1 \mathrm{X_{1t}} + \cdots + \beta_\mathrm{n} \mathrm{X_{nt}})$$

To illustrate regression models for count time series, we use the (Bike Sharing Demand dataset from Kaggle), which contains daily records of bike rentals along with features such as temperature, humidity, and holiday indicators. The model is fitted and evaluated in `R`.

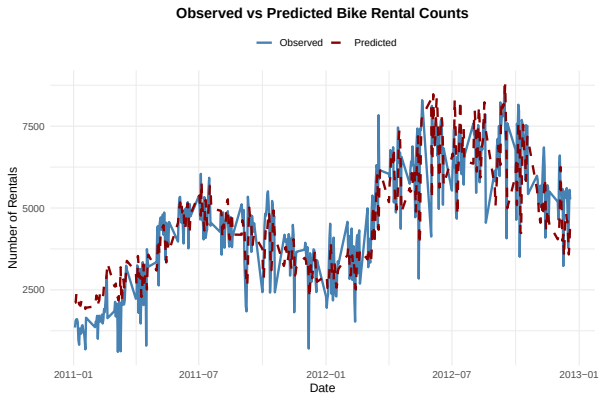


Fig. 4: Observed vs Predicted Bike Rental Counts

State Space Models

State space models describe a time series using latent (unobserved) variables that evolve. These models consist of two equations:

- **State equation:** Describes how the hidden state evolves over time.
- **Observation equation:** Relates the observed data to the hidden state.

A basic linear Gaussian state space model is given by:

$$\mathrm{x_t} = \mathrm{F x_{t-1}} + \mathrm{w_t}, \quad \mathrm{w_t} \sim \mathcal{N}(0, \mathrm{Q})$$

$$\mathrm{y_t} = \mathrm{H x_t} + \mathrm{v_t}, \quad \mathrm{v_t} \sim \mathcal{N}(0, \mathrm{R})$$

To illustrate state space modelling, we use a real dataset from Kaggle containing daily reservoir water levels in India. We estimate the hidden true water level using a local level state space model:

$$\mathrm{x_t} = \mathrm{x_{t-1}} + \mathrm{w_t}$$
$$\mathrm{y_t} = \mathrm{x_t} + \mathrm{v_t}$$

We implement this in `R` using the `dlm` package and apply the Kalman filter for smoothing.

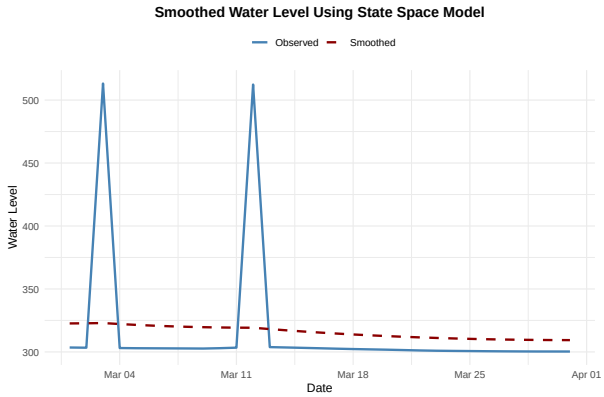


Fig. 5: Smoothed Water Level Using State Space Model

4. Methodology

Study Area

The region at the heart of our study is the **Vu Gia – Thu Bồn catchment area**, located in central Vietnam and covering the provinces of **Quảng Nam and Đà Nẵng**. This basin is formed by two major rivers: the **Vu Gia**, originating in the northwestern highlands, and the **Thu Bồn**, flowing from the central mountainous areas towards the southeast. These rivers converge before reaching the coastal plain, ultimately discharging into the South China Sea near Hội An.

This region plays a vital role in the socio-economic development of central Vietnam. It is home to a diverse range of ecosystems and supports key agricultural activities, including rice cultivation, aquaculture, and fruit farming. The basin also encompasses rapidly urbanizing zones, notably around Đà Nẵng and Hội An, which serve as important hubs for tourism and commerce.

The Vu Gia – a dense and intricate network of mountain tributaries characterizes Thu Bồn basin. Due to the region's tropical monsoon climate, it experiences intense and seasonal rainfall, making it particularly vulnerable to flooding. These floods can have devastating impacts on local communities, damaging crops, disrupting transportation, and threatening water infrastructure and livelihoods.

In our study, we focus on seven stations in particular: **Thành Mỹ, Hội Khách, Ái Nghĩa, Nông Sơn, Giao Thủy, Câu Lâu, and Hội An**. These stations allow us to track both upstream and downstream variations in water levels, which is crucial for understanding and predicting flood dynamics in the catchment.

These stations were selected because they are located in proximity to each other, resulting in a **strong correlation** between the water levels at each station.

This correlation is reinforced by the direction of the river, which flows from west to east, passing through the Thành Mỹ station and eventually emptying into the sea at the Hội An station. The location of the stations is shown on the map Figure 6.

We have also illustrated the graphs used for the **GNN and hybrid models**, this graph was built using the geographical layout of the river and the relative positions of the hydrological stations. Reflects the actual flow direction and how stations are connected along the river.

From this graph, we derived the following directed adjacency matrix:

$$\text{Adjacency Matrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

This asymmetric matrix encodes the upstream-to-downstream relationships between stations, where a value of 1 indicates a directed edge from one station to another.

Dataset

We train and validate our models using only a subset of the dataset, specifically covering the period from **January 1st, 2010, to December 31st, 2020**. This time frame was chosen to better capture recent trends in water levels while preserving the **stationarity** of the time series.

Including older data, such as from 1970 onward, could introduce **non-stationarities** due to long-term shifts and evolving hydrological patterns.

The selected data segment includes:

- **Water level**: **4,005** daily measurements
- **Rainfall**: **4,005** daily measurements

These measurements are available for **seven stations**. Since each station provides both water level and rainfall data, the complete dataset has a shape of $(4005, 14)$ where each row corresponds to a daily timestamp, and each column represents one of the 14 variables (2 variables $\times$ 7 stations).

The following graphs show the average annual water levels and rainfall for each station Figure 8, Figure 9.

Our goal is to predict the water level at a specific **target station**. To do so, we use the following 13 input features

- Water level and rainfall data from the 6 other stations (12 features)
- Rainfall value at the target station (1 feature)

This results in a total of 13 input variables used to estimate the output: the water level at the target station.

Models Architecture

CNN

The **CNN** models take as input a tensor X of shape (window size, 13) with 20 rows representing the last 20 days of time series measurements and 13 columns representing the 13 parameters used to predict the water level at the target station.

The convolution part comprises two one-dimensional convolution functions with input and output dimensions of (13, 64) and (64, 64), respectively, and a kernel size of 3 x 1.

Each convolutional layer is followed by a ReLU (*Rectified Linear Unit*) activation function, defined as:

$$\text{ReLU(x)} = \max(0, \text{x})$$

which introduces non-linearity into the model and enables it to learn more complex representations and a Flatten function is then used to convert the output of the second convolution into a two-dimensional tensor.

This tensor then enters two fully connected layers: the first layer comprises 640 neurons, followed by another ReLU activation function; the second layer comprises 50 neurons.

The final output of the network is an integer representing the predicted water level at the target station.

Transformer

The **Transformer** model takes the same tensor as the **CNN** model as input and consists of a first linear transform layer with input and output dimensions of (13, 64).

This is followed by a transform encoder using four heads to carry out the transformation in two successive layers. After this series of transformations, there are two more fully connected layers in the same form as the **CNN**.

GNN

The **GNN** model uses graph structures to learn spatial dependencies and how each node interacts with the others. This model is relevant to our project because the geography of the river, for which we wish to predict the water level, can be represented by a graph. In this graph, each station is a node and the edges represent the distances between stations.

The **GNN** input is a three-dimensional tensor with shape (**window size**, **number of stations**, 2) with 20 rows representing the last 20 days of time series measurements; 7 columns representing the seven different stations; and a depth of 2 representing the two parameters for each station: **water level and precipitation**. This input can be seen as a reshaped version of the **CNN/Transformer** input.

However, since $2 \times 7 = 14$ and not 13, an adjustment is needed. To maintain consistency with the original 13-feature structure, we exclude the water level of the target station (the prediction target) from the input. Consequently, during reshaping, we insert a placeholder (for example, a column of zeros) into the corresponding position to preserve the required (window size, 7, 2) shape.

This allows us to match the **GNN** input format without introducing target leakage.This representation characterises each node of the graph using the water level and rainfall measurements from the last 20 days.

The **GNN** model comprises two successive GCN convolution functions with input and output dimensions of (7,64) and (64,64), respectively, with a ReLU activation function interposed between them.

The suite comprises three fully connected layers, with 448, 128, and 64 neurons respectively. The first two layers are interposed with a ReLU activation function.

Series Model (GNN + Transformer Hybrid)

The **Series** model uses the same tensors as the **GNN** model and consists of a **GNN** and a transformer architecture linked together. This combination aims to exploit the strengths of both models.

The architecture of our models is summarised in Figure 7, which provides a visual representation of the four models described above.

The complete implementation of our models can be accessed on GitHub at:
https://github.com/Rassoul24/Stage-Vietnam

Preprocessing and Training

The following steps were applied: To train our models, we applied the following steps: We normalised the data using Min-Max scaling to ensure uniformity between variables. Min-Max Scaling (also known as normalization) is a preprocessing technique used to rescale the values of a variable to a fixed range, typically $[0, 1]$. This method is particularly useful for machine learning models that are sensitive to the scale of input data, such as neural networks and support vector machines.

Formula. Given a variable X, the Min-Max Scaling is defined as:

$$X_{\text{scaled}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \tag{2}$$

where:

- X is the original value,
- $X_{\min}$ and $X_{\max}$ are the minimum and maximum values observed in the feature,
- X_{scaled} is the rescaled value in the range $[0, 1]$.

Remarks:.

- This method preserves the shape of the original distribution.
- It is sensitive to outliers, which can skew the scaling range.

We used the **Mean Squared Error (MSE)** as the loss function, since our task involves the prediction of continuous variables. The MSE is defined as:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

where y_i is the true value, $\hat{y}_i$ is the predicted value, and n is the total number of samples. This loss function penalizes large errors more heavily, encouraging the model to make accurate predictions.

To minimize this loss, we used the **Adam** optimizer (Adaptive Moment Estimation), which combines the advantages of both the AdaGrad and RMSProp algorithms. The update rule for the parameters θ at each iteration t is given by:

$$\theta_t = \theta_{t-1} - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon}$$

where $\hat{m}_t$ and $\hat{v}_t$ are the bias-corrected estimates of the first and second moments of the gradients, α is the learning rate, and ε is a small constant to prevent division by zero.

This combination allows the optimizer to adapt the learning rate for each parameter, leading to faster and more stable convergence during training.

For consistency and interpretability, we used these hyperparameters to train our models:

Hyperparameter	Value	Description
window_size	20	Number of pastime steps used as input to predict future values.
batch_size	32	Number of samples processed before the model is updated.
epochs	50	Number of complete passes through the training dataset.
learning_rate	0.001	Step size used by the optimizer to update model weights.

Tabla 1: Model hyperparameters and their descriptions.

We experimented with different values of the **window_size** parameter and decided to fix it at 20, as it provided the best overall performance for our models.

Our model's loss function converges to zero, showing that they learn correctly with these hyperparameters as we can this in the Figure 15.

We used 90% of our dataset for training and reserved the remaining 10% for validation. To ensure the reproducibility of our results, we fixed the random seed to 42 when splitting the data. This decision was motivated by the observation that the performance of our models varied significantly depending on how the training set was randomly partitioned. By setting a fixed seed, we eliminate this source of randomness and ensure consistent and comparable results across experiments.

Evaluation Metric

The following metrics were used to evaluate model performance:

- **Mean Absolute Error (MAE)**: Measures the average absolute difference between predicted and true values.
- **Mean Squared Error (MSE)**: Measures average squared difference; sensitive to larger errors.
- **Root Mean Squared Error (RMSE)**: Square root of MSE; interpretable in same units as the target.
- **R-squared Score** (R^2): Indicates the proportion of the variance in the target variable that is predictable from the input features.

5. Experiments

In our experiments, we first calculate the correlation between the Target variable and the Features, finding a very strong correlation between the water levels at the different stations and the target variable however, the rain variable was found to have a weak impact on the target variable as we can see in the Figure 10. Because of this strong correlation we decided to train a multivariable linear regression model as a statistical baseline. We then compare the following deep learning models:

Tabla 2: Evaluation Metric Definitions

Metric	Formula	Interpretation
MAE	$\frac{1}{n}\sum \lvert y_i - \hat{y}_i \rvert$	Average absolute error
MSE	$\frac{1}{n}\sum (y_i - \hat{y}_i)^2$	Penalizes large errors
RMSE	$\sqrt{\text{MSE}}$	Error magnitude in target unit
R^2	$1 - \frac{\sum (y_i - \hat{y}_i)^2}{\sum (y_i - \bar{y})^2}$	Variance explained by the model

- **CNN** – Captures short-term spatial patterns in time series.
- **Transformer** – Models long-range dependencies using self-attention mechanisms.
- **GNN** – Encodes spatial-temporal relationships through graph-based structures.
- **Series Model (Proposed)** – A hybrid architecture combining GNN and Transformer components to leverage their complementary strengths.

6. Experimental Results

We plot our results over the rainy season, from September to December, as this period is crucial for flood prediction. To enhance the interpretability of the results and provide better insight into flood potential, we include the annual mean water level as well as the mean water level during the rainy season.

Let's begin with the results of our linear model. As shown in Figure 11, the model performs well, which can be attributed to the strong linear correlation between the input variables and the target. This confirms that the underlying relationships in the data are largely linear.

The linear model also exhibits an extremely fast training time — only 0.01 seconds — which makes it highly efficient computationally, as indicated in Table 3.

When compared to our deep learning models, we observe that the **CNN** also delivers strong performance, with a notably low **MSE** and a high R^2 score, indicating its ability to effectively capture relevant patterns in the data. The training time is still low with 22.51 seconds as indicated in Table 3.

It performs particularly well in predicting low water levels, especially in the range of approximately 8 to 10 meters (shown in Figure 13 a).

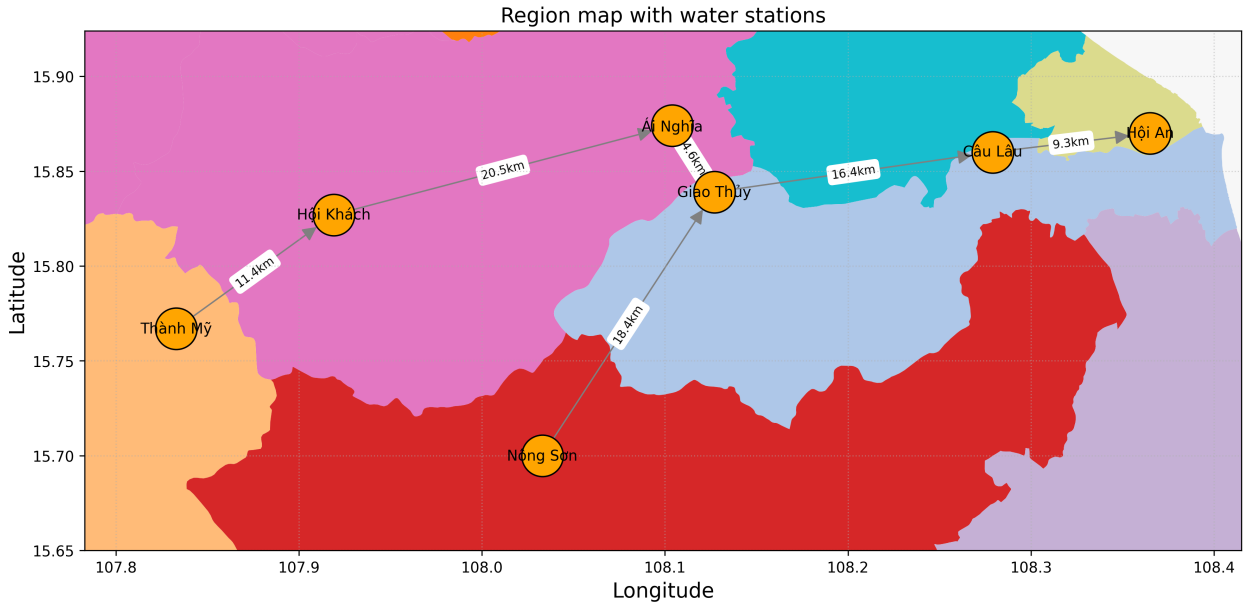


Fig. 6: Region Map with Water Stations.

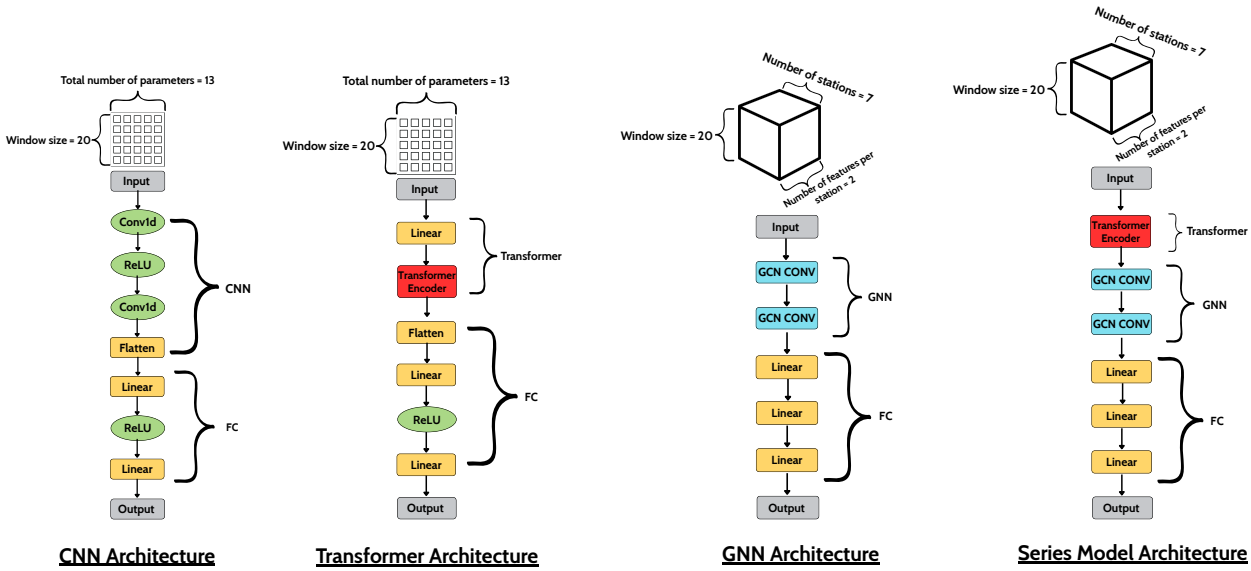


Fig. 7: Model Architectures

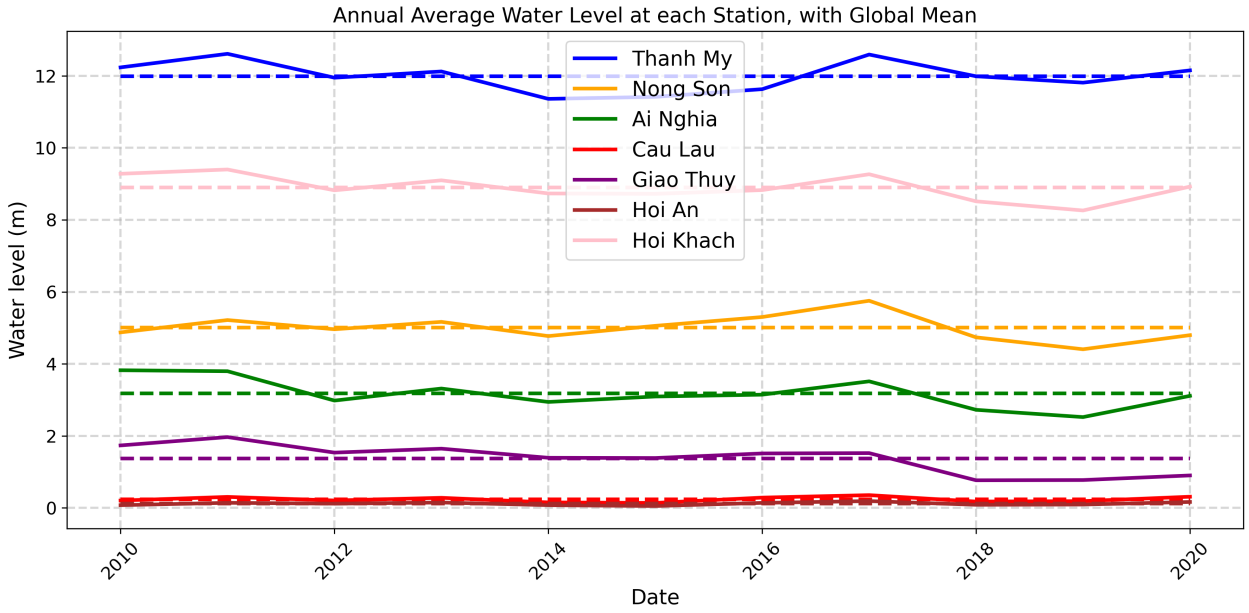


Fig. 8: Annual Average Water Level at Each Station, with Global Mean.

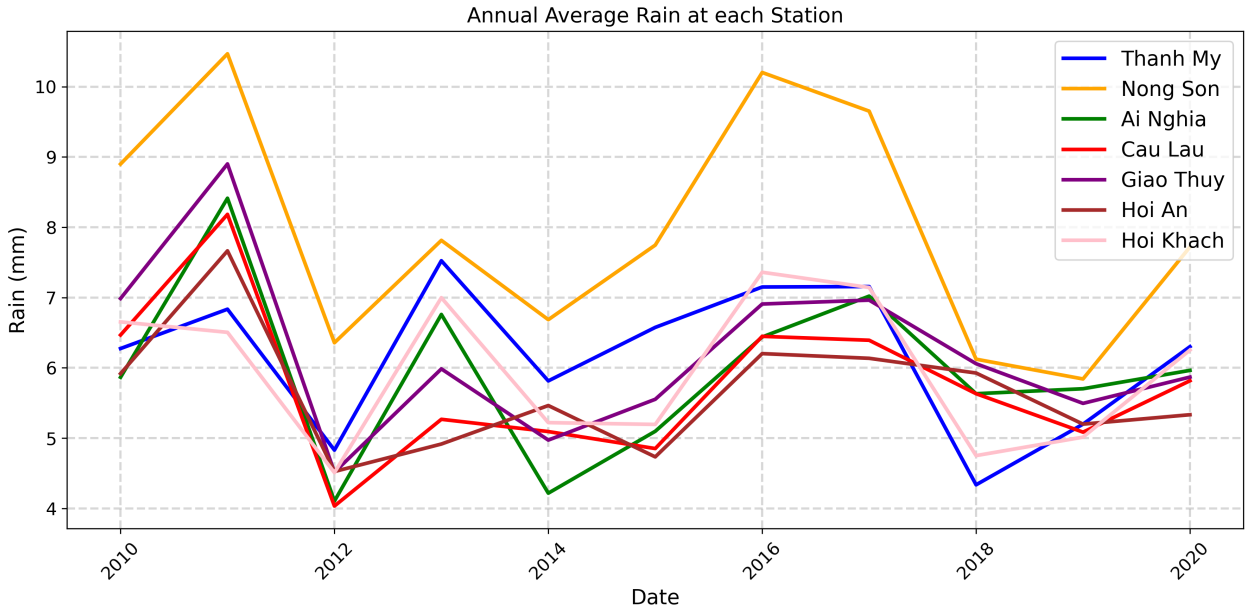


Fig. 9: Annual Average Rain at Each Station.

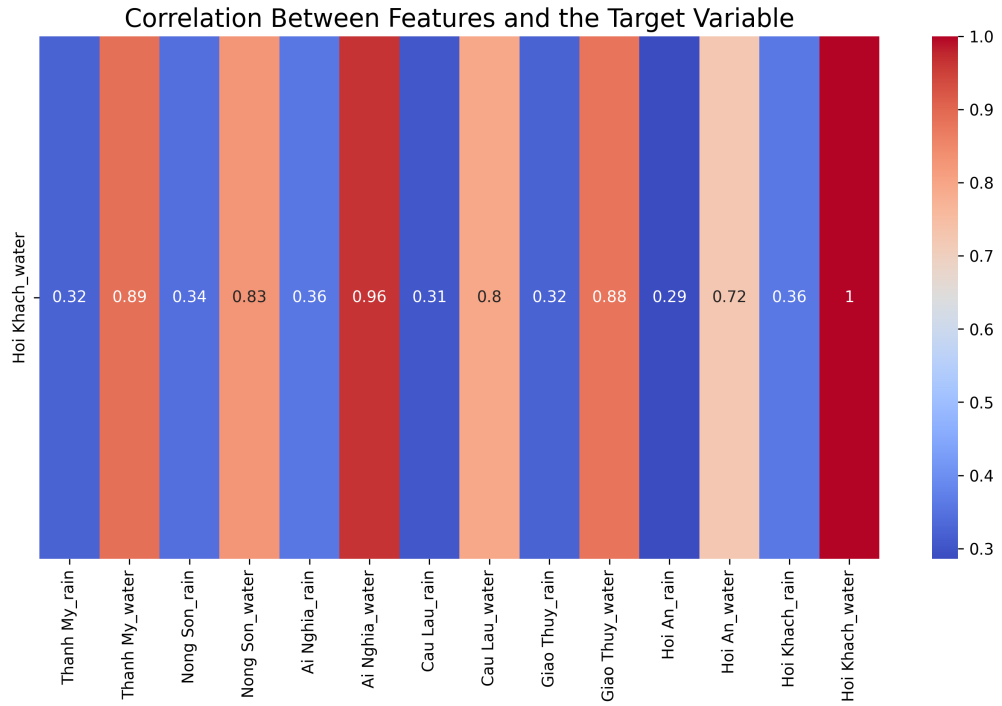


Fig. 10: Correlation Between Features and the Target Variable.

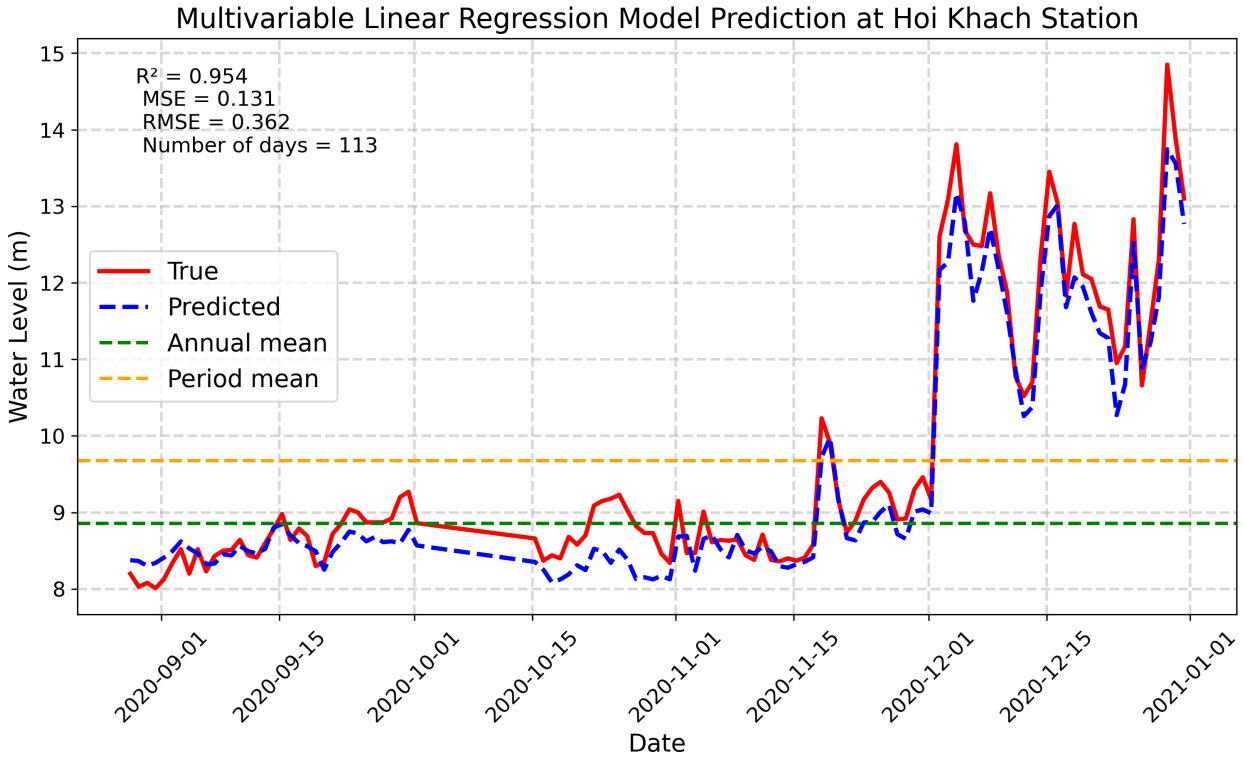


Fig. 11: Linear Regression Model Prediction at Hoi Khach Station.

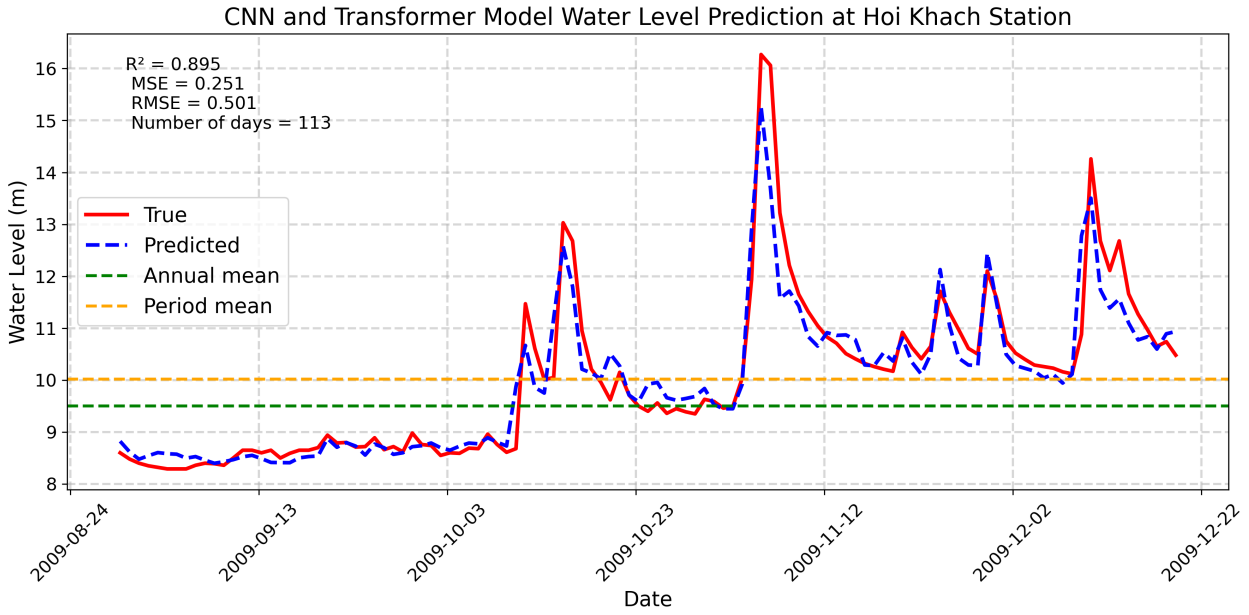
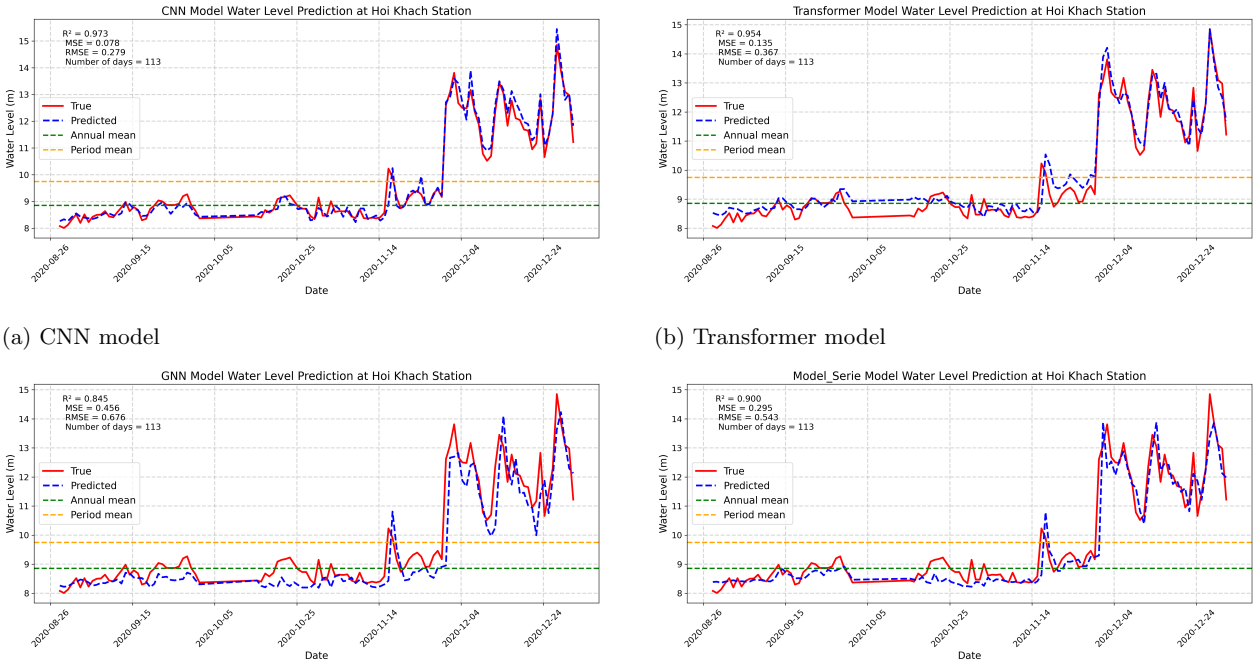


Fig. 12: Predicted Water Levels using the Mean ensemble of CNN and Transformer outputs (2009 season).



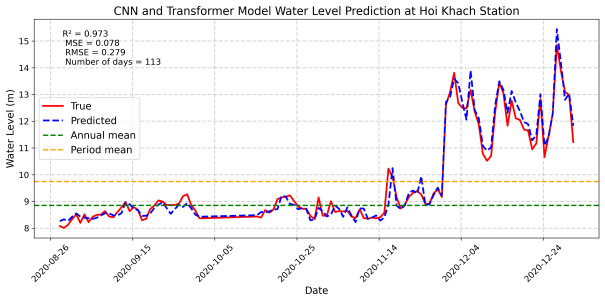
(a) CNN model

(b) Transformer model

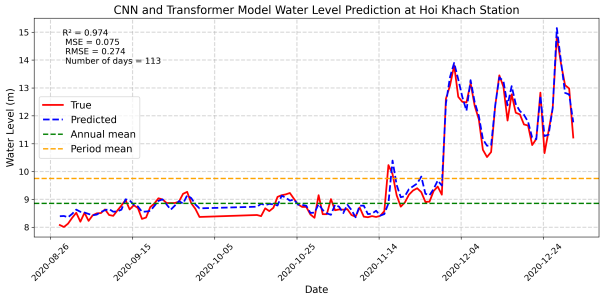
(c) GNN model (Asymmetric Graphe)

(d) Sequential Hybrid model

Fig. 13: Predicted water levels using four different models: (a) CNN, (b) Transformer, (c) GNN, and (d) Hybrid Sequential.

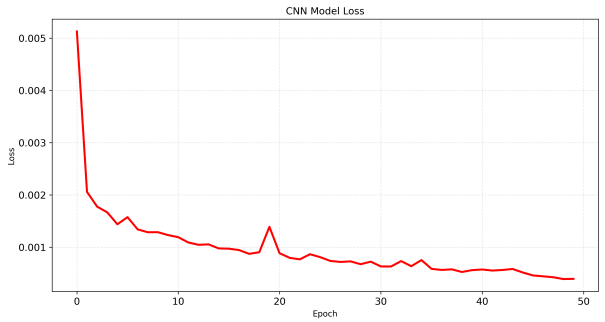


(a) Threshold-based ensemble of CNN and Transformer outputs



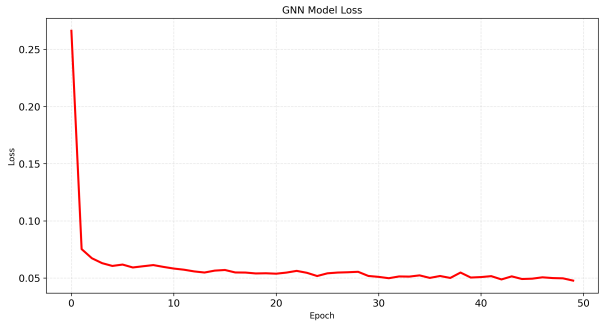
(b) Mean ensemble of CNN and Transformer outputs

Fig. 14: Comparison between two modelling strategies for Water Level Prediction.

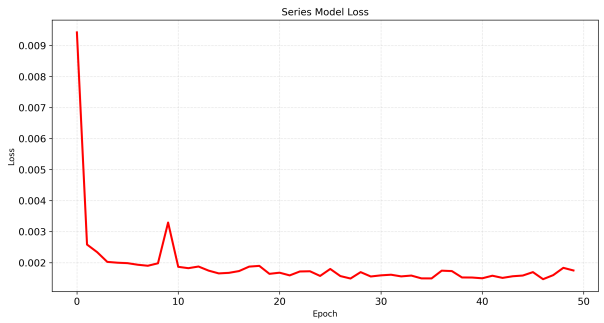


(a) CNN model loss

(b) Transformer model loss



(c) GNN model loss



(d) Sequential Hybrid model loss

Fig. 15: Loss function for different models: (a) CNN, (b) Transformer, (c) GNN, and (d) Hybrid Sequential.

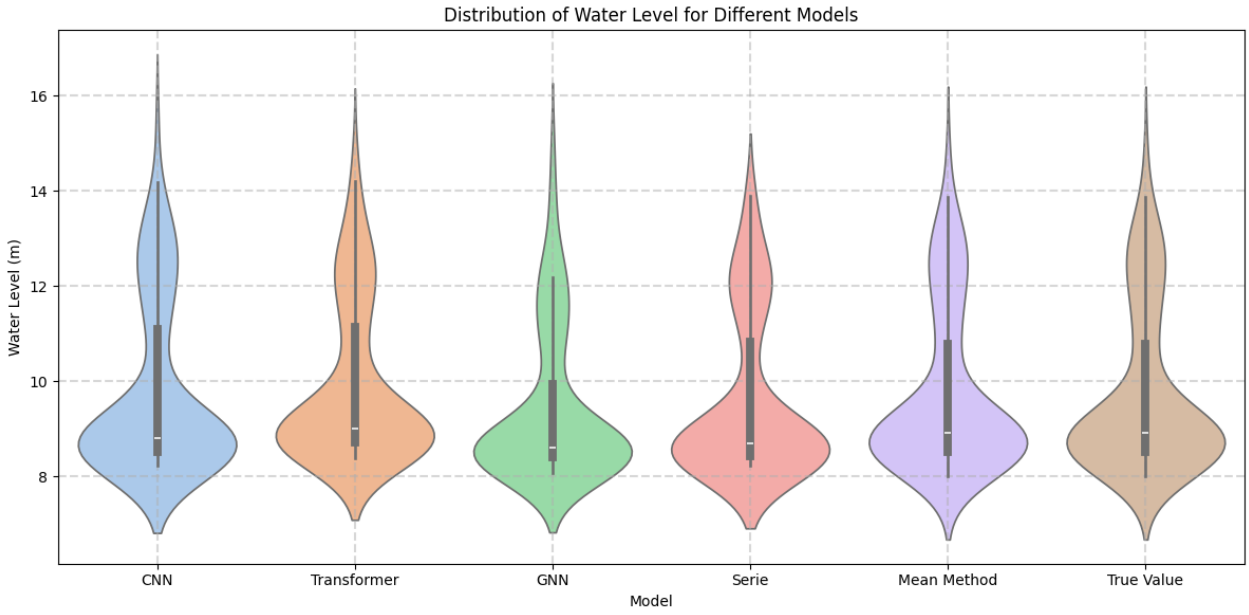


Fig. 16: Distribution of Water Level for Different Models.

Tabla 3: Model performance comparison (↓ = lower is better, ↑ = higher is better)

Model	**Time (s) ↓**	**MSE ↓**	**RMSE ↓**	**MAE ↓**	R^2 ↑
Linear Regression	0.01	0.0785	0.2801	0.2206	0.9534
CNN	20.51	0.0779	0.2792	0.1975	0.9735
Transformer	442.03	0.1350	0.3675	0.2857	0.9540
CNN + Transformer (Thre-based)	462.54	0.0779	0.2792	**0.1975**	0.9735
CNN + Transformer (Mean-based)	462.54	**0.0751**	**0.2740**	0.2035	**0.9744**
GNN (Symmetric Graph)	416.77	0.4694	0.6851	0.4550	0.8402
GNN (Asymmetric Graph)	402.79	0.4563	0.6755	0.4658	0.8446
GNN + Transformer	13107.97	0.2951	0.5432	0.3411	0.8995

The results show that the **Transformer** model performs well, as evidenced by its low MSE and high R^2 score. It is particularly effective at predicting high water levels, especially those exceeding 10 meters.

Notably, the model accurately captures peak water levels, which is essential for reliable flood forecasting (see Figure 13 b). We also observe a significant increase in training time compared to the **CNN** model, rising from 22.51 to 442.03 seconds — a substantial proportional growth.

The **GNN** model provides less accurate predictions compared to the **CNN** and **Transformer** models, as reflected by a higher **MSE** and a lower R^2 score (see in Figure 13 c). The **GNN** model with an asymmetric graph outperforms the one with a symmetric graph (shown in Table 3). This can be attributed to the fact that the asymmetric graph more accurately captures the natural directionality of river flow. The training time remains comparable to that of the **Transformer** model.

The Series model is less accurate than the **CNN** and **Transformer** models, but outperforms the **GNN**, as Furthermore, it has the highest training time among all models, reaching 13107.97 seconds — a significant difference compared to the **CNN** model (shown in Table 3).

We made an empirical observation: The **CNN** model performs better for low water level predictions, while the **Transformer** achieves higher accuracy when predicting high water levels, as previously stated. Based on this insight, we designed a hybrid prediction strategy that leverages the strengths of both models.

Specifically, if the last predicted water level exceeds a certain threshold, the **Transformer** model is used to generate the next prediction; otherwise, the **CNN** model is applied.

A threshold of 10 meters was chosen, as this approximately corresponds to the onset of water level rise, typically observed in early December.This sequential combination leads to improved accuracy (shown in Figure 14 a).

In addition, we experimented with a simple ensemble

approach, where the final prediction is computed as the average of the **CNN** and **Transformer** outputs.

$$\text{Output} = \frac{\text{CNN} + \text{T}}{2}$$

This method also yielded promising results. We observe an improvement in prediction results with a R^2 score of **0.9744**, which is particularly valuable for reliable flood forecasting (shown in Figure 14 b).

Once all models have been trained and evaluated, we can compare them using the metrics defined earlier. The results presented in Table 3 show that the best performing model is the **CNN + Transformer (Mean-based)** ensemble, which achieves the lowest **MSE** and the highest R^2 score.

Moreover, the violin plot in Figure 16 illustrates that the **CNN + Transformer (Mean-based)** model provides the most consistent and precise predictions. The distribution is tightly centred around the true values, with low variance and few extreme deviations, indicating high reliability and robustness.

Interestingly, this simpler ensemble outperforms the more complex model **GNN + Transformer**, which integrates both temporal and spatial information.

This may be attributed to the limited size of the data set (2010-2020), which may not be sufficient for the **GNN-based** models to fully capture spatial dependencies and relationships between stations.

GNNs are likely to perform better with larger datasets or when enriched with additional topographic and hydrological features.

To assess whether our model overfits, we tested it on a 2009 rainy season, which lies outside the training dataset. The Figure 12 shows the predicted water levels by applying the mean ensemble of the **CNN and Transformer** outputs.

We obtain the following evaluation metrics for this method :

- **MSE**: 0.1565
- **RMSE**: 0.3957
- **MAE**: 0.2620
- **R^2 Score**: 0.9342

These results confirm that the model maintains strong predictive performance even on unseen data. In particular, the high R^2 score of 0.9342 suggests that the model generalizes well and does not suffer from overfitting.

7. Conclusion

Our experiments demonstrate that it is possible to obtain highly accurate results using various deep learning models. Among the four architectures tested, the **CNN** model proved to be the most effective, combining high predictive performance ($R^2 = 0.9735$) with low training time (only 20.51 seconds).

Compared to the **linear regression** model, the **CNN** significantly improves accuracy, raising the R^2 score **from 0.9534 to 0.9765**, which highlights the benefit of using deep learning approaches—even when the data exhibits strong linear dependencies.

Building on these results, we combined the strengths of our models and introduced the **CNN + Transformer (mean-based)** model, which yielded the best overall performance. Interestingly, despite being structurally simpler than the GNN and other sequence-based models, it achieved superior results.

This outcome can be explained by the nature of our dataset and the underlying graph structure. Specifically, the relatively small dataset size and the use of a simple distance-based adjacency matrix may have limited the expressiveness of more complex models like **GNNs**. Moreover, the dataset may not contain sufficient non-linear patterns to justify the use of high-capacity architectures.

We have developed models capable of making highly accurate predictions using only a few parameters and with reduced computational time. These models also take advantage of the geographical specificities of our study area.

While our current work focuses on short-term water level forecasting using historical observations, future research could explore the integration of rainfall forecasts to extend the prediction horizon. Such improvements could contribute to more robust early-warning systems for flood risk management.

References

[1] Shi, J., Stebliankin, V., Wang, Z., Wang, S., & Narasimhan, G. (2023). *Graph Transformer Network for flood forecasting with heterogeneous covariates. arXiv preprint*, arXiv:2310.07631. https://doi.org/10.48550/arXiv.2310.07631

[2] Murray, C. J., Du Bois, N., Hollywood, L., & Coyle, D. (2023). *State-of-the-art deep learning models are superior for time series forecasting and are applied optimally with iterative prediction methods. SSRN.* https://doi.org/10.2139/ssrn.4361707

[3] Bui, D. T., Hoang, N. D., Nguyen, H., *et al.* (2020). *A novel deep learning neural network approach for predicting flash flood susceptibility: A case study at a high frequency tropical storm area. Science of The Total Environment*, 701, 134413. https://doi.org/10.1016/j.scitotenv.2019.134413

[4] Nhu, O. L., Thuy, N. T. T., Wilderspin, I., & Coulier, M. *A preliminary analysis of flood and storm disaster data in Viet Nam.*

[5] Chen, C., Jiang, J., Liao, Z., Zhou, Y., Wang, H., & Pei, Q. (2022). *A short-term flood prediction based on spatial deep learning network: A case study for Xi County, China. Journal of Hydrology*, 607, 127535. https://doi.org/10.1016/j.jhydrol.2022.127535

[6] Berkhahn, S., Fuchs, L., & Neuweiler, I. (2019). *An ensemble neural network model for real-time prediction of urban floods. Journal of Hydrology*, 575, 743–754. https://doi.org/10.1016/j.jhydrol.2019.05.066

[7] Overpeck, J. T., Meehl, G. A., Bony, S., & Easterling, D. R. (2011). *Climate data challenges in the 21st century. Science*, 331(6018), 700–702. https://doi.org/10.1126/science.1197869

[8] Adamowski, J., Chan, H. F., Prasher, S. O., Ozga-Zielinski, B., & Sliusarieva, A. (2012). *Comparison of multiple linear and nonlinear regression, autoregressive integrated moving average, artificial neural network, and wavelet artificial neural network methods for urban water demand forecasting in Montreal, Canada. Water Resources Research*, 48(1), W01528. https://doi.org/10.1029/2010WR009945

[9] Chen, C., Hui, Q., Xie, W., Wan, S., Zhou, Y., & Pei, Q. (2021). *Convolutional neural networks for forecasting flood process in Internet-of-Things enabled smart city. Computer Networks*, 186, 107744. https://doi.org/10.1016/j.comnet.2020.107744

[10] Casolaro, A., Capone, V., Iannuzzo, G., & Camastra, F. (2023). *Deep learning for time series forecasting: Advances and open problems. Information*, 14(11), 598. https://doi.org/10.3390/info14110598

[11] Mosavi, A., Ozturk, P., & Chau, K. (2018). *Flood prediction using machine learning models: Literature review. Water*, 10(11), 1536. https://doi.org/10.3390/w10111536

[12] Hu, R., Fang, F., Pain, C. C., & Navon, I. M. (2019). *Rapid spatio-temporal flood prediction and uncertainty quantification using a deep learning method. Journal of Hydrology*, 575, 911–920. https://doi.org/10.1016/j.jhydrol.2019.05.087

[13] Fokianos, K., & Kedem, B. (2003). *Regression theory for categorical time series. Statistical Science*, 18(3). https://doi.org/10.1214/ss/1076102425

[14] Patterson, T., Thomas, L., Wilcox, C., Ovaskainen, O., & Matthiopoulos, J. (2008). *State–space models of individual animal movement. Trends in Ecology & Evolution*, 23(2), 87–94. https://doi.org/10.1016/j.tree.2007.10.009

[15] Weiß, C. H. (2021). *Stationary count time series models. Wiley Interdisciplinary Reviews: Computational Statistics*, 13(1), e1502. https://doi.org/10.1002/wics.1502

[16] World Bank. (2021). *Vietnam Country Risk Profile: Flood.* Climate Change Knowledge Portal. https://climateknowledgeportal.worldbank.org/country/vietnam/vulnerability

[17] Kratzert, F., Klotz, D., Brenner, C., Schulz, K., & Herrnegger, M. (2018). *Rainfall–runoff modelling using Long Short-Term Memory (LSTM) networks.* Hydrology and Earth System Sciences, 22(11), 6005–6022. https://doi.org/10.5194/hess-22-6005-2018

[18] Box, G.E.P., Jenkins, G.M., Reinsel, G.C., & Ljung, G.M. (2015). *Time Series Analysis: Forecasting and Control* (5th ed.). John Wiley & Sons. https://onlinelibrary.wiley.com/doi/book/10.1002/9781118619193

[19] Hyndman, R.J., & Athanasopoulos, G. (2018). *Forecasting: Principles and Practice* (2nd ed.). OTexts. https://otexts.com/fpp2/

[20] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[21] Gers, F.A., Schmidhuber, J., & Cummins, F. (2000). Learning to forget: Continual prediction with LSTM. *Neural Computation*, 12(10), 2451–2471. https://doi.org/10.1162/089976600300015015

[22] Bai, S., Kolter, J.Z., & Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modelling. *arXiv preprint* arXiv:1803.01271. https://arxiv.org/abs/1803.01271

[23] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems* (Vol. 30, pp. 5998–6008). https://arxiv.org/abs/1706.03762

[24] Li, S., Jin, X., Xuan, Y., Zhou, X., Chen, W., Wang, Y.X., & Yan, X. (2019). Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In *NeurIPS 32*. https://papers.nips.cc/paper/2019/hash/31f8b9d6b8ba73b676f0ea33eac63e37-Abstract.html

[25] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Yu, P.S. (2020). A comprehensive survey on graph neural networks. *IEEE Transactions*

on Neural Networks and Learning Systems, 32(1), 4–24. https://doi.org/10.1109/TNNLS.2020.2978386

[26] Jiang, J., Zhang, Y., Liu, Q., & Wang, P. (2022). Graph neural networks for multivariate time series forecasting: A survey. *IEEE Transactions on Neural Networks and Learning Systems*. https://doi.org/10.1109/TNNLS.2021.3139483

[27] Lütkepohl, H. (2005). *New Introduction to Multiple Time Series Analysis*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-27752-1

[28] Zhang, Y., Liu, Y., Qin, H., & Wang, Y. (2019). Hybrid models for short-term traffic flow prediction using deep learning and statistical methods. *Transportation Research Part C: Emerging Technologies*, 101, 1–16. https://doi.org/10.1016/j.trc.2019.01.008