Electric Power Systems Research 255 (2026) 112754

Contents lists available at ScienceDirect

Electric Power Systems Research

journal homepage: www.elsevier.com/locate/epsr

Sparse mixture of experts enhanced transformer architecture for short-term hydroelectric reservoir volume prediction

Laio Oriel Seman [a,*], Kin-Choong Yow [b], Stefano Frizzo Stefenon [b,c]

[a] Department of Automation and Systems Engineering, Federal University of Santa Catarina, Florianópolis, Brazil
[b] Faculty of Engineering and Applied Sciences, University of Regina, Saskatchewan, S4S 0A2, Canada
[c] Lisbon School of Engineering (ISEL), Polytechnic University of Lisbon (IPL), Lisbon, Portugal

ARTICLE INFO

Keywords:
Electrical power system
Hydroelectric reservoir
Transformer architecture
Mixture-of-experts

ABSTRACT

In hydroelectric-based systems, effective energy generation planning relies heavily on precise forecasting of reservoir water levels. This paper proposes a novel hybrid forecasting framework that integrates multiple preprocessing strategies with a sparse Mixture of Experts enhanced Transformer architecture for short-term reservoir volume prediction. When evaluated on 19 interconnected reservoirs across two major river basins in southern Brazil using real operational data from the Brazilian National System Operator, the proposed model achieves a mean squared error of 0.062 and a mean absolute error of 0.145. Comprehensive benchmarking against 18 state-of-the-art deep learning methods demonstrates that the proposed approach significantly outperforms existing methods while maintaining computational efficiency through sparse expert routing. Our results confirm that combining diverse preprocessing strategies with conditional computation mechanisms provides superior forecasting accuracy for reservoir management in hydroelectric power systems.

1. Introduction

Hydroelectric power constitutes a significant portion of Brazil's total electricity generation capacity, thus serving as the basis of the national electricity system [1]. The stable and efficient operation of this extensive hydroelectric infrastructure depends critically on effective reservoir management, as many reservoirs are hydrologically and operationally interconnected. One of the primary challenges in this context lies in the accurate prediction of reservoir useful volume, which directly influences the generation capacity and overall system reliability [2].

The useful volume of a reservoir, defined as its available operational storage, is a fundamental metric for assessing water availability and power generation potential [3]. Conventional forecasting methods for reservoir volume have predominantly relied on classical time series and statistical models [4]. Although such techniques provide useful insights, they often fall short in representing complex spatiotemporal dependencies. These dependencies arise from interactions among multiple interconnected reservoirs. This limitation becomes particularly pronounced in densely coupled hydroelectric systems, leading to suboptimal forecasting performance [5].

Within this context, the southern region of Brazil offers a particularly relevant and challenging case study. This region comprises a network of 19 reservoirs distributed across the Uruguay and Iguaçu river basins. These reservoirs operate under strong hydraulic interdependence. In this network, upstream reservoir operations exert direct influence on downstream dynamics. This highlights the need for predictive models capable of capturing both spatial and temporal correlations inherent to such interconnected hydroelectric systems [6]. The interconnectedness, combined with the temporal dynamics of water inflow, precipitation, and operational decisions, creates a complex spatiotemporal forecasting problem. Such problems demand advanced modeling approaches.

Recent advances in deep learning have demonstrated the effectiveness of Transformer architectures for modeling long-range dependencies in time series data [7]. Notable examples include the Temporal Fusion Transformer (TFT) [8], Patch Time Series Transformer (PatchTST) [9], Decomposable Multiscale Mixing for Time Series Forecasting (TimeMixer) [10], Temporal 2D-Variation Modeling for General Time Series Analysis (TimesNet) [11], and Inverted Transformer for Time Series Forecasting (iTransformer) [12]. Considering the vast nomenclature used in the field, Table 1 presents the list of acronyms used in this paper.

Standard Transformer models face challenges when applied to multivariate time series. These challenges arise from heterogeneous temporal patterns and high computational demands. To address these limitations, this paper proposes a novel hybrid forecasting framework. The framework integrates multiple preprocessing strategies with a sparse Mixture

* Corresponding author.
E-mail addresses: laio.seman@ufsc.br (L.O. Seman), Kin-Choong.Yow@uregina.ca (K.-C. Yow), stefano.stefefenon@isel.br (S.F. Stefenon).

https://doi.org/10.1016/j.epsr.2026.112754
Received 25 October 2025; Received in revised form 18 December 2025; Accepted 12 January 2026
Available online 21 January 2026

Table 1
List of acronyms used.

Full name	Acronym
Adaptive Moment Estimation	Adam
Average Megawatts	MWmed
Bidirectional Temporal Convolutional Network	BiTCN
Brazilian National System Operator	ONS
Channel-Aligned Robust Blend Transformer	CARD
Convolutional Neural Network	CNN
Deep Autoregressive Network	DeepAR
Decomposable Multiscale Mixing for Time Series Forecasting	TimeMixer
Differentiable Architecture Search	DARTS
Empirical Wavelet Transform	EWT
Energy Natural Affluent	ENA
Feed-Forward Network	FFN
Gated Recurrent Unit	GRU
Inverted Transformer for Time Series Forecasting	iTransformer
Long Short-Term Memory	LSTM
Mean Absolute Error	MAE
Mean Absolute Percentage Error	MAPE
Mean Squared Error	MSE
Mixture of Experts	MoE
Multi-Head Attention	MHA
Multi-Resolution Time-Series Transformer	MTST
Multi-Scale Convolution	MSConv
Multiseasonal Trend Decomposition using LOESS	MSTL
Neural Architecture Search	NAS
Neural Basis Expansion Analysis for Time Series	NBEATS
Neural Hierarchical Interpolation Time Series	NHITS
Patch Time Series Transformer	PatchTST
Rectified Linear Unit	ReLU
Reversible Instance Normalization	RevIN
Root Mean Square Normalization	RMSNorm
Root Mean Squared Error	RMSE
Spectral Temporal Graph Neural Network	StemGNN
Temporal 2D-Variation Modeling for General Time Series Analysis	TimesNet
Temporal Convolutional Network	TCN
Temporal Fusion Transformer	TFT
Temporal Irregularity-aware Dynamic Encoding	TiDE

of Experts (MoE) enhanced Transformer architecture for short-term (2-h ahead) reservoir volume prediction.

The proposed model architecture consists of three main components. First, a multi-head preprocessing module applies parallel feature extraction strategies. These strategies include reversible instance normalization, seasonal-trend decomposition, multi-scale convolutions, and patch embedding to capture diverse temporal characteristics. Second, a Transformer encoder augmented with sparse MoE layers enables conditional computation and improved representational capacity while maintaining computational efficiency. Third, an autoregressive Transformer decoder with masked self-attention and cross-attention mechanisms enables multi-horizon forecasting.

Considering the aspects of the proposed model, this paper has the following contributions:

- A novel multi-head preprocessing framework that captures diverse temporal characteristics through parallel feature extraction. The framework includes reversible instance normalization for handling distributional shifts, seasonal-trend decomposition for separating long-term and periodic patterns, multi-scale convolutions for multi-resolution feature extraction, and patch embedding for efficient sequence tokenization.
- A sparse MoE enhanced Transformer architecture designed specifically for short-term reservoir volume prediction. The architecture incorporates conditional computation through expert routing to improve model capacity while maintaining computational efficiency. It also includes load-balancing mechanisms to ensure stable training dynamics.
- An extensive evaluation using real operational data from 19 interconnected reservoirs across two major river basins in southern Brazil from the Brazilian National System Operator (ONS), demonstrating the model's effectiveness in practical scenarios.
- A comprehensive comparative analysis against 18 state-of-the-art forecasting methods. These include recent Transformer variants, classical deep learning architectures, and specialized time series models. The analysis provides insights into the performance characteristics and computational trade-offs of different modeling approaches for reservoir forecasting.

The remainder of this paper is organized as follows: Section 2 presents our methodology, including the multi-head preprocessing framework, MoE-enhanced Transformer encoder, autoregressive decoder architecture, and training objectives. The experimental setup, performance measures, and dataset characteristics are presented in Section 3. Section 4 presents experimental results, preprocessing analysis, and comprehensive benchmarking against state-of-the-art methods. Finally, Section 5 concludes the paper and discusses future research directions.

2. Methodology

We propose a comprehensive framework for multivariate time series forecasting that integrates adaptive signal preprocessing with neural architecture search. The framework consists of two main components operating at different stages of the prediction pipeline: (1) **signal denoising and adaptive preprocessing**, which extracts temporal patterns using EWT and applies learnable transformations through a differentiable HeadComposer module, and (2) **transformer-based prediction**, comprising an attention-based encoder-decoder architecture with sparse Mixture-of-Experts (MoE) computation. These components are jointly

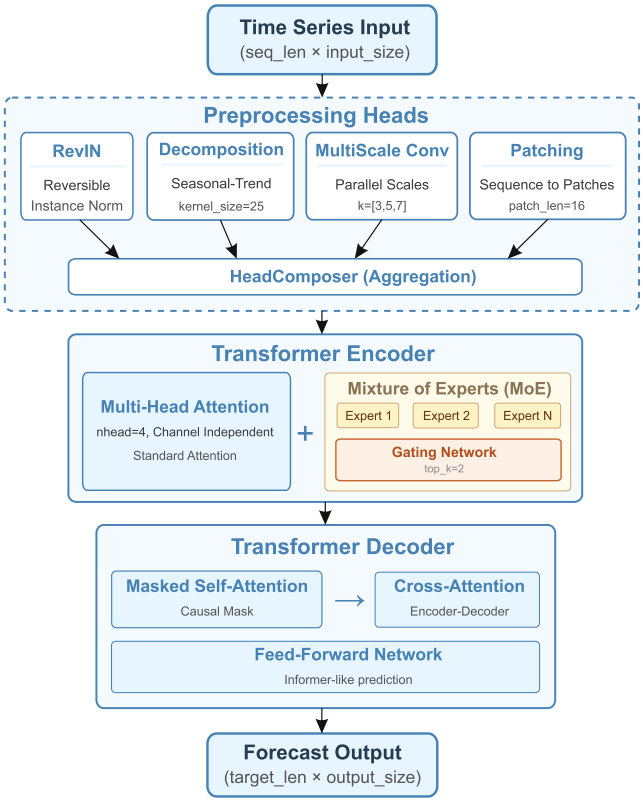

Fig. 1. Architecture of the proposed method integrating EWT-based signal denoising, differentiable HeadComposer for preprocessing selection, and transformer encoder-decoder with sparse MoE layers.

optimized through a bilevel optimization procedure, as illustrated in Fig. 1.

2.1. Problem formulation

Given a multivariate time series $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_{L_x}]^\top \in \mathbb{R}^{L_x \times d_x}$ of length L_x with d_x input features, our objective is to forecast the subsequent L_y time steps $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_{L_y}]^\top \in \mathbb{R}^{L_y \times d_y}$. We assume the observations are generated by an underlying stochastic process:

$$\mathbf{Y} = f^\star(\mathbf{X}) + \epsilon, \qquad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}), \tag{1}$$

where $f^\star$ denotes the unknown data-generating functional and $\mathbf{\Sigma}$ is the noise covariance matrix.

Unlike fixed-pipeline approaches, our framework jointly optimizes both network weights $\boldsymbol{\theta}$ and preprocessing architecture parameters $\boldsymbol{\alpha}$ via bilevel optimization:

$$\boldsymbol{\alpha}^\star = \arg\min_{\boldsymbol{\alpha}} \ \mathcal{L}_{\text{val}}(\boldsymbol{\theta}^\star(\boldsymbol{\alpha}), \boldsymbol{\alpha}), \quad \text{s.t.} \quad \boldsymbol{\theta}^\star(\boldsymbol{\alpha}) = \arg\min_{\boldsymbol{\theta}} \ \mathcal{L}_{\text{train}}(\boldsymbol{\theta}, \boldsymbol{\alpha}), \tag{2}$$

where $\mathcal{L}_{\text{train}}$ and $\mathcal{L}_{\text{val}}$ denote the training and validation losses, respectively. This formulation enables end-to-end adaptation of both preprocessing structure and model parameters.

2.2. Stage I: signal denoising and adaptive preprocessing

The first stage focuses on extracting meaningful temporal patterns and applying adaptive transformations to the input time series. This stage consists of two components: EWT for signal denoising, and the HeadComposer module for differentiable preprocessing selection.

2.2.1. Empirical wavelet transform for denoising

The Empirical Wavelet Transform, short EWT, is applied to perform denoising on time series data. The EWT procedure begins by dividing the signal's Fourier spectrum into contiguous frequency intervals through the identification of local maxima, which reveal the locations of dominant frequency components [13].

After spectrum partitioning, empirical wavelets are generated as bandpass filters corresponding to each frequency interval. These filters extract signal features within their respective frequency bands, yielding a decomposition into intrinsic modes [14]. The empirical scaling function $\hat{\phi}_n(\omega)$ and wavelet function $\hat{\psi}_n(\omega)$ are defined as:

$$\hat{\phi}_n(\omega) = \begin{cases} 1, & \text{if } |\omega| \le \omega_n - \tau_n, \\ \cos\left(\frac{\pi}{2}\beta\left(\frac{|\omega| - \omega_n + \tau_n}{2\tau_n}\right)\right), & \text{if } \omega_n - \tau_n \le |\omega| \le \omega_n + \tau_n, [6pt] \\ 0, & \text{otherwise,} \end{cases} \tag{3}$$

$$\hat{\psi}_n(\omega) = \begin{cases} 1, & \text{if } \omega_n + \tau_n \le |\omega| \le \omega_{n+1} - \tau_{n+1}, \\ \cos\left(\frac{\pi}{2}\beta\left(\frac{|\omega| - \omega_{n+1} + \tau_{n+1}}{2\tau_{n+1}}\right)\right), & \text{if } \omega_{n+1} - \tau_{n+1} \le |\omega| \le \omega_{n+1} + \tau_{n+1}, \\ \sin\left(\frac{\pi}{2}\beta\left(\frac{|\omega| - \omega_n + \tau_n}{2\tau_n}\right)\right), & \text{if } \omega_n - \tau_n \le |\omega| \le \omega_n + \tau_n, \\ 0, & \text{otherwise,} \end{cases} \tag{4}$$

where $\beta(x)$ is an arbitrary function satisfying $\beta(x) = 0$ if $x \le 0$, $\beta(x) = 1$ if $x \ge 1$, and $\beta(x) + \beta(1 - x) = 1$ for all $x \in [0, 1]$.

The EWT coefficients are computed as $W_f^E(n, t) = \langle f, \psi_n \rangle$, and the signal is reconstructed as:

$$f(t) = W_f^E(0, t) * \phi_1(t) + \sum_{n=1}^{N} W_f^E(n, t) * \psi_n(t), \tag{5}$$

yielding empirical modes $f_k(t) = W_f^E(k, t) * \psi_k(t)$ that capture distinct frequency components. The denoised signal $\tilde{\mathbf{X}}$ is obtained by discarding high-frequency modes associated with noise.

2.2.2. HeadComposer: differentiable preprocessing selection

Following EWT denoising, the HeadComposer module enables differentiable neural architecture search (NAS) over a library of preprocessing transformations. Let $\mathcal{H} = \{h_1, \ldots, h_N\}$ denote the set of available preprocessing heads, where each head h_i transforms an input sequence, potentially altering temporal resolution or feature dimensionality. The HeadComposer adaptively selects or attenuates transformations based on dataset characteristics, eliminating the need for manual preprocessing design.

Motivation from time series properties. Time series in environmental and hydrological domains frequently exhibit characteristics that challenge forecasting models:

- **Non-stationarity:** Shifts in mean and variance over time, induced by seasonal cycles or long-term trends, cause discrepancies between training and test distributions [15].
- **Additive temporal components:** Many time series admit decomposition into trend, seasonal, and residual components.
- **Multi-scale temporal patterns:** Informative patterns emerge at varying temporal resolutions, from short-term fluctuations to long-term cycles.
- **Local temporal coherence:** Consecutive time steps exhibit high similarity, supporting aggregation of local windows into higher-level representations [16].

Reservoir volume time series typically exhibit these properties: fluctuations in mean and variance arise from seasonal precipitation and operational policies; pronounced annual cycles and multi-year trends align

with explicit decomposition; dynamics operate across timescales from daily inflows to seasonal cycles; and gradual day-to-day changes favor patch-based representations.

Sequential processing architecture. All preprocessing heads are applied sequentially in a fixed order, forming a transformation pipeline where each head's contribution is controlled by a learnable architecture parameter. Given the denoised signal $\tilde{\mathbf{X}} = \mathbf{Z}^{(0)}$, the pipeline computes:

$$\mathbf{Z}^{(0)} \xrightarrow{h_1} \mathbf{Z}^{(1)} \xrightarrow{h_2} \mathbf{Z}^{(2)} \xrightarrow{h_3} \mathbf{Z}^{(3)} \xrightarrow{h_4} \mathbf{Z}^{(4)} = \mathbf{H}, \tag{6}$$

where the output $\mathbf{H}$ serves as input to the transformer encoder. This sequential design allows transformations to compose and build upon each other, with earlier operations (e.g., normalization) influencing later ones (e.g., patch embedding).

Architecture parameterization. Each head h_i is associated with a scalar architecture parameter $\alpha_i \in \mathbb{R}$ that controls how the head output is combined with its input. We employ two parameterization modes depending on the transformation's properties:

Gate Mode (Discrete Selection). For heads that alter sequence length or require a well-defined inverse (e.g., patch embedding, reversible normalization), a binary gate is employed using a straight-through estimator:

$$g_i = \mathbb{I}[\sigma(\alpha_i) > 0.5] + \sigma(\alpha_i) - \mathrm{sg}[\sigma(\alpha_i)], \tag{7}$$

where $\sigma(\cdot)$ is the sigmoid function and $\mathrm{sg}[\cdot]$ denotes the stop-gradient operator. The transformation becomes:

$$\mathbf{Z}^{(i)} = g_i \cdot h_i(\mathbf{Z}^{(i-1)}) + (1 - g_i) \cdot \mathbf{Z}^{(i-1)}. \tag{8}$$

Soft Mode (Continuous Blending). For dimensionality-preserving heads, a soft combination enables smooth interpolation:

$$w_i = \sigma(\alpha_i), \qquad \mathbf{Z}^{(i)} = w_i \cdot h_i(\mathbf{Z}^{(i-1)}) + (1 - w_i) \cdot \mathbf{Z}^{(i-1)}. \tag{9}$$

Preprocessing head library. The framework includes the following preprocessing transformations, applied in the order listed:

(1) **Reversible Instance Normalization (RevIN).** RevIN [15] performs instance-wise normalization along the temporal dimension for each variable independently. For input $\mathbf{Z}^{(0)} \in \mathbb{R}^{L_x \times d_x}$, we compute per-channel statistics:

$$\mu_c = \frac{1}{L_x} \sum_{t=1}^{L_x} Z_{t,c}^{(0)}, \qquad \sigma_c = \sqrt{\frac{1}{L_x} \sum_{t=1}^{L_x} (Z_{t,c}^{(0)} - \mu_c)^2}, \tag{10}$$

and apply normalization:

$$\mathrm{RevIN}(\mathbf{Z}^{(0)})_{t,c} = \frac{Z_{t,c}^{(0)} - \mu_c}{\sigma_c + \epsilon}, \quad \epsilon = 10^{-6}. \tag{11}$$

The statistics (μ_c, σ_c) are stored and later used to denormalize predictions, recovering the original scale:

$$\hat{Y}_{t,c} = \hat{Z}_{t,c} \cdot (\sigma_c + \epsilon) + \mu_c. \tag{12}$$

RevIN is controlled via **gated** architecture parameter α_{revin}.

(2) **Seasonal–Trend Decomposition.** A moving-average filter with kernel size $k = 25$ decomposes the signal into trend and seasonal components:

$$\mathbf{Z}_{\text{trend}} = \mathrm{AvgPool1D}(\mathbf{Z}^{(1)}; k), \qquad \mathbf{Z}_{\text{seasonal}} = \mathbf{Z}^{(1)} - \mathbf{Z}_{\text{trend}}. \tag{13}$$

The outputs are concatenated along the feature dimension and projected back to dimension d_x via a linear layer. The result is blended with the input using **soft mode** parameterization with weight α_{decomp}.

(3) **Multi-Scale Convolution (MSConv).** Parallel 1D convolutions with kernel sizes $\mathcal{S} = \{3, 5, 7, 11\}$ capture multi-scale temporal patterns:

$$\mathrm{MSConv}(\mathbf{Z}^{(2)}) = \mathrm{Concat}\left[\sigma(\mathrm{Conv}(\mathbf{Z}^{(2)}; s))\right]_{s \in \mathcal{S}}, \tag{14}$$

where $\sigma(\cdot)$ denotes ReLU activation. The concatenated result is projected back to d_x channels via a linear layer and blended using **soft** architecture parameter α_{msconv}.

(4) **Patch Embedding.** Following PatchTST [16], the sequence is divided into non-overlapping patches of fixed length p:

$$\mathbf{P}_i = \mathbf{Z}^{(3)}_{(i-1)p+1:ip,:} \in \mathbb{R}^{p \times d_x}, \quad i = 1, \ldots, N_p = \lfloor L_x / p \rfloor. \tag{15}$$

Each patch is flattened and linearly projected to the model embedding dimension d_h:

$$\mathbf{h}_i = \mathbf{W}_p \,\mathrm{vec}(\mathbf{P}_i) + \mathbf{b}_p, \tag{16}$$

where $\mathbf{W}_p \in \mathbb{R}^{d_h \times (p \cdot d_x)}$ and $\mathbf{b}_p \in \mathbb{R}^{d_h}$ are learnable parameters. This operation reduces the effective sequence length by a factor of p, balancing computational efficiency with temporal resolution. Patch embedding is controlled via **gated** parameter α_{patch}.

The final output $\mathbf{H} = \mathbf{Z}^{(4)} \in \mathbb{R}^{L_h \times d_h}$ serves as the input representation to the transformer encoder, where L_h depends on whether patch embedding is activated.

2.3. Stage II: transformer-based prediction

The second stage consists of an attention-based transformer encoder-decoder architecture augmented with sparse conditional computation through Mixture-of-Experts layers. The preprocessed embeddings from the HeadComposer serve as input to this neural architecture.

2.3.1. Positional encoding

To imbue the preprocessed embeddings $\mathbf{H}$ with temporal ordering information, we incorporate positional encodings additively:

$$\mathbf{H}^{(0)} = \mathbf{H} + \mathbf{PE}, \tag{17}$$

where $\mathbf{PE} \in \mathbb{R}^{L_h \times d_h}$ employs sinusoidal encodings:

$$\mathrm{PE}_{t,2i} = \sin\left(\frac{t}{10000^{2i/d_h}}\right), \qquad \mathrm{PE}_{t,2i+1} = \cos\left(\frac{t}{10000^{2i/d_h}}\right), \tag{18}$$

enabling the model to capture relative and absolute positional information.

2.3.2. Encoder architecture

The encoder transforms the positionally-encoded representation $\mathbf{H}^{(0)} \in \mathbb{R}^{L_h \times d_h}$ into a contextualized latent sequence $\mathbf{Z}_e \in \mathbb{R}^{L_h \times d_h}$ through L_e stacked layers. Each layer consists of a multi-head self-attention module followed by a sparse Mixture-of-Experts feed-forward block, with residual connections and layer normalization applied after each sub-module.

Multi-head self-attention. For an input representation $\mathbf{H}^{(\ell)} \in \mathbb{R}^{L_h \times d_h}$ at layer ℓ, we compute queries, keys, and values through learned linear transformations:

$$\mathbf{Q} = \mathbf{H}^{(\ell)}\mathbf{W}_Q, \quad \mathbf{K} = \mathbf{H}^{(\ell)}\mathbf{W}_K, \quad \mathbf{V} = \mathbf{H}^{(\ell)}\mathbf{W}_V, \tag{19}$$

where $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{d_h \times d_k}$. The scaled dot-product attention mechanism is defined as:

$$\mathrm{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}. \tag{20}$$

To enhance representational capacity, we employ r parallel attention heads operating on different learned subspaces:

$$\mathrm{MHA}(\mathbf{H}^{(\ell)}) = \left[\mathrm{Attn}_1 \| \ldots \| \mathrm{Attn}_r\right]\mathbf{W}_O, \tag{21}$$

where $\|$ denotes concatenation and $\mathbf{W}_O \in \mathbb{R}^{rd_k \times d_h}$ is the output projection matrix.

Each encoder layer applies:

$$\tilde{\mathbf{H}}^{(\ell)} = \mathrm{RMSNorm}(\mathbf{H}^{(\ell-1)}), \tag{22}$$

$$\mathbf{H}_1^{(\ell)} = \mathbf{H}^{(\ell-1)} + \mathrm{MHA}(\tilde{\mathbf{H}}^{(\ell)}), \tag{23}$$

$$\mathbf{H}^{(\ell)} = \mathbf{H}_1^{(\ell)} + \mathrm{MoE}(\mathrm{RMSNorm}(\mathbf{H}_1^{(\ell)})), \tag{24}$$

where RMSNorm is defined as:

$$\mathrm{RMSNorm}(\mathbf{x}) = \frac{\mathbf{x}}{\sqrt{\frac{1}{d_h}\sum_{i=1}^{d_h} x_i^2 + \epsilon}} \odot \boldsymbol{\gamma}, \quad \epsilon = 10^{-5}, \tag{25}$$

with learned scale parameter $\boldsymbol{\gamma} \in \mathbb{R}^{d_h}$.

Mixture of experts feed-forward network. Following recent advances in sparse conditional computation [17,18], we replace the standard dense feed-forward network with a MoE layer. The MoE consists of E expert networks $\{E_j\}_{j=1}^{E}$, each parameterized as a two-layer fully connected network:

$$E_j(\mathbf{h}_t) = \mathbf{W}_{2,j}\,\sigma(\mathbf{W}_{1,j}\mathbf{h}_t + \mathbf{b}_{1,j}) + \mathbf{b}_{2,j}, \tag{26}$$

where $\mathbf{W}_{1,j} \in \mathbb{R}^{d_{\mathrm{ff}} \times d_h}$, $\mathbf{W}_{2,j} \in \mathbb{R}^{d_h \times d_{\mathrm{ff}}}$, $\sigma(\cdot)$ is the ReLU activation, and d_{ff} is the intermediate dimension.

A lightweight gating network produces routing scores over experts:

$$\mathbf{r} = \mathrm{softmax}(\mathbf{W}_g\mathbf{h}_t + \mathbf{b}_g) \in \mathbb{R}^E, \tag{27}$$

where $\mathbf{W}_g \in \mathbb{R}^{E \times d_h}$ and $\mathbf{b}_g \in \mathbb{R}^E$. For computational efficiency, we employ top-k routing, selecting only the k experts with highest gating probabilities:

$$\mathrm{MoE}(\mathbf{h}_t) = \sum_{j \in \mathcal{T}_k(\mathbf{h}_t)} \bar{r}_j \cdot E_j(\mathbf{h}_t), \qquad \bar{r}_j = \frac{r_j}{\sum_{j' \in \mathcal{T}_k} r_{j'}}, \tag{28}$$

where $\mathcal{T}_k(\mathbf{h}_t)$ denotes the indices of the top-k experts and $\bar{r}_j$ are the renormalized gating weights.

To prevent expert collapse where few experts receive the majority of tokens, an auxiliary load-balancing loss promotes uniform utilization:

$$\mathcal{L}_{\mathrm{aux}} = \lambda_{\mathrm{aux}} \cdot E \sum_{j=1}^{E} f_j \cdot P_j, \tag{29}$$

where $f_j = \frac{1}{BL_h}\sum_{b,t} \mathbb{1}[j \in \mathcal{T}_k(\mathbf{h}_{b,t})]$ is the fraction of tokens routed to expert j, $P_j = \frac{1}{BL_h}\sum_{b,t} r_j(\mathbf{h}_{b,t})$ is the mean gating probability, B is the batch size, and $\lambda_{\mathrm{aux}} > 0$ is a balancing coefficient.

2.3.3. Decoder architecture

The decoder generates predictions autoregressively through L_d stacked layers. It is initialized with $\mathbf{D}_{\mathrm{in}} = \mathrm{Concat}[\mathbf{Y}_{\mathrm{label}}, \mathbf{Y}_{\mathrm{zero}}] \in \mathbb{R}^{(L_{\mathrm{label}}+L_y)\times d_y}$, where $\mathbf{Y}_{\mathrm{label}}$ contains the last L_{label} observed values from $\mathbf{X}$ and $\mathbf{Y}_{\mathrm{zero}}$ is zero-padded for the forecast horizon. Each decoder layer contains masked self-attention, encoder-decoder cross-attention, and feed-forward sub-modules.

Masked self-attention. To preserve the autoregressive property and prevent information leakage from future time steps, causal masking is applied:

$$\mathbf{A}_{\mathrm{mask}} = \mathrm{softmax}\left(\frac{\mathbf{Q}_d\mathbf{K}_d^\top}{\sqrt{d_k}} + \mathbf{M}\right)\mathbf{V}_d, \tag{30}$$

where $\mathbf{M} \in \mathbb{R}^{L_y \times L_y}$ is a causal mask with $M_{ij} = 0$ if $i \geq j$ and $M_{ij} = -\infty$ otherwise.

Cross-attention. Cross-attention enables the decoder to selectively attend to relevant portions of the encoded input sequence. At layer ℓ, queries are derived from the decoder state $\mathbf{S}^{(\ell-1)}$, while keys and values originate from the encoder output $\mathbf{Z}_e$:

$$\mathrm{CrossAttn}(\mathbf{S}^{(\ell-1)}, \mathbf{Z}_e) = \mathrm{softmax}\left(\frac{(\mathbf{S}^{(\ell-1)}\mathbf{W}_Q^c)(\mathbf{Z}_e\mathbf{W}_K^c)^\top}{\sqrt{d_k}}\right)(\mathbf{Z}_e\mathbf{W}_V^c). \tag{31}$$

Each decoder layer applies:

$$\tilde{\mathbf{S}}^{(\ell)} = \mathrm{RMSNorm}(\mathbf{S}^{(\ell-1)}), \tag{32}$$

$$\mathbf{S}_1^{(\ell)} = \mathbf{S}^{(\ell-1)} + \mathrm{MaskedMHA}(\tilde{\mathbf{S}}^{(\ell)}), \tag{33}$$

$$\mathbf{S}_2^{(\ell)} = \mathbf{S}_1^{(\ell)} + \mathrm{CrossAttn}(\mathrm{RMSNorm}(\mathbf{S}_1^{(\ell)}), \mathbf{Z}_e), \tag{34}$$

$$\mathbf{S}^{(\ell)} = \mathbf{S}_2^{(\ell)} + \mathrm{FFN}(\mathrm{RMSNorm}(\mathbf{S}_2^{(\ell)})), \tag{35}$$

where FFN is a standard two-layer feed-forward network with ReLU activation.

Output projection. The final decoder output is projected to produce multivariate forecasts:

$$\hat{\mathbf{Y}} = \mathrm{RMSNorm}(\mathbf{S}^{(L_d)})\mathbf{W}_o + \mathbf{b}_o \in \mathbb{R}^{L_y \times d_y}, \tag{36}$$

where $\mathbf{W}_o \in \mathbb{R}^{d_h \times d_y}$ and $\mathbf{b}_o \in \mathbb{R}^{d_y}$. If RevIN was activated during preprocessing, the stored statistics are used to denormalize predictions.

Scheduled sampling. To bridge the gap between teacher-forcing training and autoregressive inference, decoder inputs are sampled as:

$$\mathbf{d}_t = \begin{cases} y_{t-1} & \text{w.p. } \rho(\epsilon), \\ \hat{y}_{t-1} & \text{w.p. } 1-\rho(\epsilon), \end{cases} \qquad \rho(\epsilon) = \max(0, 0.95 - \epsilon/E_{\mathrm{total}}), \tag{37}$$

gradually exposing the model to its own predictions as training progresses.

2.4. Bilevel optimization

Model training follows a bilevel optimization scheme that decouples learning of network weights from optimization of preprocessing architecture parameters, as illustrated in Fig. 2.

2.4.1. Weight optimization (inner level)

With architecture parameters $\boldsymbol{\alpha}$ fixed, model weights $\boldsymbol{\theta}$ are optimized on the training set by minimizing:

$$\mathcal{L}_{\mathrm{train}}(\boldsymbol{\theta}, \boldsymbol{\alpha}) = \mathbb{E}_{(\mathbf{X},\mathbf{Y})\sim\mathcal{D}_{\mathrm{train}}}\left[\ell(\mathbf{Y}, f(\mathbf{X}; \boldsymbol{\theta}, \boldsymbol{\alpha})) + \mathcal{L}_{\mathrm{aux}}(\boldsymbol{\theta})\right], \tag{38}$$

where $\ell(\cdot,\cdot)$ denotes the MSE loss:

$$\ell(\mathbf{Y}, \hat{\mathbf{Y}}) = \frac{1}{L_y d_y}\sum_{t=1}^{L_y} \|\hat{\mathbf{y}}_t - \mathbf{y}_t\|_2^2. \tag{39}$$

Optimization employs AdamW with learning rate $\eta_\theta = 10^{-4}$, weight decay $\lambda_{\mathrm{wd}} = 10^{-2}$, and exponential moving average coefficients $\beta_1 = 0.9$, $\beta_2 = 0.999$. Gradient clipping with maximum norm 1.0 stabilizes training.

2.4.2. Architecture optimization (outer level)

After a warmup period of $E_{\mathrm{warmup}} = 5$ epochs, architecture parameters $\boldsymbol{\alpha} = \{\alpha_{\mathrm{revin}}, \alpha_{\mathrm{decomp}}, \alpha_{\mathrm{msconv}}, \alpha_{\mathrm{patch}}\}$ are optimized while keeping $\boldsymbol{\theta}$ fixed. The objective is defined on the validation set:

$$\mathcal{L}_{\mathrm{val}}(\boldsymbol{\theta}, \boldsymbol{\alpha}) = \mathbb{E}_{(\mathbf{X},\mathbf{Y})\sim\mathcal{D}_{\mathrm{val}}}[\ell(\mathbf{Y}, f(\mathbf{X}; \boldsymbol{\theta}, \boldsymbol{\alpha}))]. \tag{40}$$

Architecture parameters are updated using Adam with learning rate $\eta_\alpha = 3 \times 10^{-3}$. For gated heads, gradients are propagated through discrete decisions using the straight-through estimator.

2.4.3. Alternating schedule

Following Differentiable Architecture Search (DARTS) [19], weight and architecture updates are alternated within each epoch after the warmup phase. For every $N_{\mathrm{alt}} = 1$ weight update step on a training mini-batch, a corresponding architecture update is performed using a mini-batch from the validation set. This alternating procedure enables joint adaptation of preprocessing structure and model parameters while maintaining stable optimization dynamics. Fig. 2 summarizes the complete training procedure.

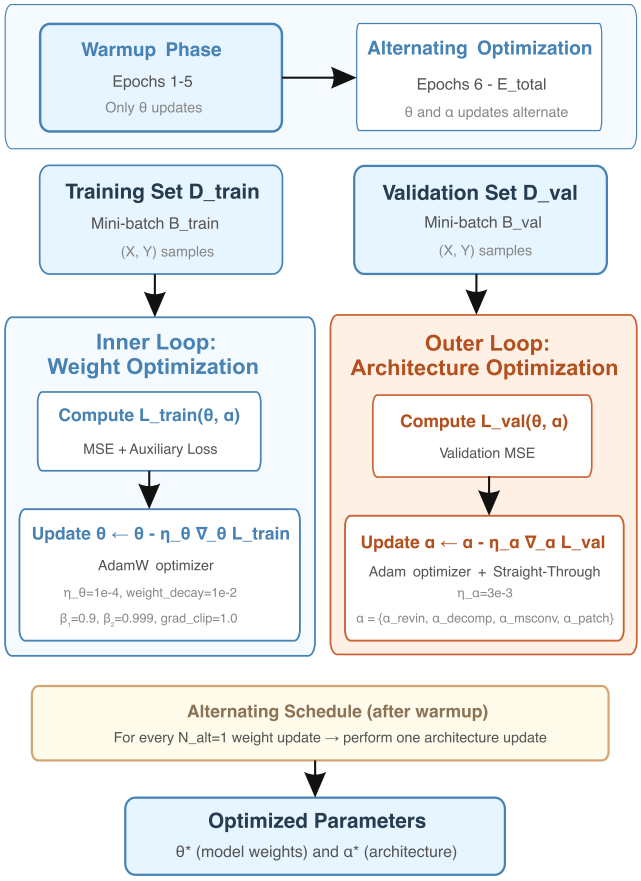

Fig. 2. Bilevel optimization procedure. Model weights $\boldsymbol{\theta}$ are optimized on training data in the inner loop, while preprocessing architecture parameters $\boldsymbol{\alpha}$ are refined on validation data in the outer loop, with intra-epoch alternation after an initial warmup phase.

Table 2
Comparison of representative time series forecasting methods.

Method	Transformer	Patch	Ch Ind	Decomp	RevIN	MoE
DLinear [25]	×	×	×	×	×	×
PatchTST [16]	✓	✓	✓	×	×	×
iTransformer [26]	✓	×	×	×	×	×
TFT [27]	✓	×	×	×	×	×
MTST [28]	✓	×	×	✓	×	×
CARD [29]	✓	×	✓	×	×	×
Time-MoE [17]	✓	×	×	×	×	✓
Moirai-MoE [18]	✓	×	×	×	×	✓
Autoformer [30]	✓	×	×	✓	×	×
RevIN [15]	×	×	×	×	✓	×
MSTL-based [31]	×	×	×	✓	×	×
TFT [32]	✓	×	×	×	×	×
CNN-LSTM [33]	×	×	×	✓	×	×
GCN-LSTM [34]	×	×	×	×	×	×
GCN-LSTM [35]	×	×	×	×	×	×
MHA-GCN [36]	✓	×	×	×	×	×
Ours	✓	✓	✓	✓	✓	✓

Channel Independence (Ch Ind), Signal Decomposition (Decomp).

2.5. Functional composition

The complete forecasting model can be expressed as a composition of the aforementioned components:

$$f_{\theta,\alpha} = \text{Denorm} \circ \text{Proj}_{\text{out}} \circ \text{Dec}_{\theta} \circ \text{Enc}_{\theta} \circ \text{PosEnc} \circ \text{HeadComposer}_{\alpha} \circ \text{EWT}, \quad (41)$$

where EWT denotes signal denoising, $\text{HeadComposer}_{\alpha}$ represents the differentiable preprocessing pipeline parameterized by $\boldsymbol{\alpha}$, PosEnc adds positional information, Enc_{θ} and Dec_{θ} denote the transformer encoder and decoder parameterized by $\boldsymbol{\theta}$, Proj_{out} is the final linear projection, and Denorm applies RevIN denormalization if activated.

2.5.1. Model configuration

Unless otherwise specified, the architecture employs the following hyperparameters: encoder depth $L_e = 1$, decoder depth $L_d = 1$, hidden dimension $d_h = 128$, number of attention heads $r = 8$, feed-forward dimension $d_{\text{ff}} = 512$, number of experts $E = 8$ with top-$k = 2$ routing, MoE balancing coefficient $\lambda_{\text{aux}} = 10^{-2}$, patch size $p = 5$, and decomposition kernel size $k = 25$. Architecture parameters are initialized as $\alpha_i = 0$ (corresponding to $\sigma(\alpha_i) = 0.5$), allowing unbiased selection during early training.

For benchmarking, we consider the TFT, PatchTST, TimeMixer, TimesNet, NHITS (Neural Hierarchical Interpolation Time Series), Long Short-Term Memory (LSTM), Neural Basis Expansion Analysis for Time Series (NBEATS), NBEATSx, Gated Recurrent Unit (GRU), Deep Autoregressive Network (DeepAR), Temporal Convolutional Network (TCN), Bidirectional TCN (BiTCN), iTransformer, TimeXer, Informer, VanillaTransformer, Spectral Temporal Graph Neural Network (StemGNN), Temporal Irregularity-aware Dynamic Encoding (TiDE), and Autoformer, considering the default hyperparameters of `neuralforecast.models` library available at `Nixtla` repository [20].

2.6. Comparison to previous approaches

Our model introduces several key novelties that distinguish it from existing Transformer-based forecasting models such as Autoformer [21], FEDformer [22], and PatchTST [23]. While these models focus on architectural innovations like decomposition mechanisms (Autoformer [21]), frequency enhancement (FEDformer [22]), or patch-based tokenization (PatchTST [23]), our proposed framework integrates a multi-head adaptive preprocessing pipeline, including EWT [24] denoising and a differentiable `HeadComposer` module that dynamically selects and combines normalization, decomposition, multi-scale convolution, and patch embedding based on data characteristics.

Our proposed approach incorporates sparse MoE layers within the Transformer encoder, enabling conditional computation that enhances representational capacity without proportional computational cost. This combination of learnable preprocessing and sparse expert routing, optimized via a bilevel training strategy, allows the model to better capture non-stationary, multi-scale, and spatially correlated patterns in reservoir data, leading to significantly lower prediction errors compared to the cited baselines.

Table 2 presents a comparison of our model to other architectures applied for time series forecasting. Here, we additionally included the Multi-Resolution Time-Series Transformer (MTST), Channel-Aligned Robust Blend Transformer (CARD), Multiseasonal Trend Decomposition using LOESS (MSTL), Convolutional Neural Network (CNN), and Multi-Head Attention (MHA) methods.

3. Experimental setup and dataset

The experimental setup was executed on a workstation using an Intel® Core™ i9-14900K processor comprising 32 cores operating at a frequency of 6.00 GHz, coupled with an NVIDIA GeForce RTX 4090 graphics card. The system was configured with 32 GB of memory and operated under Linux operating system, providing a stable environment for computational experimentation.

3.1. Performance measures

All models were developed using Python, with the training phase accelerated through the graphics processing capabilities of the RTX 4090

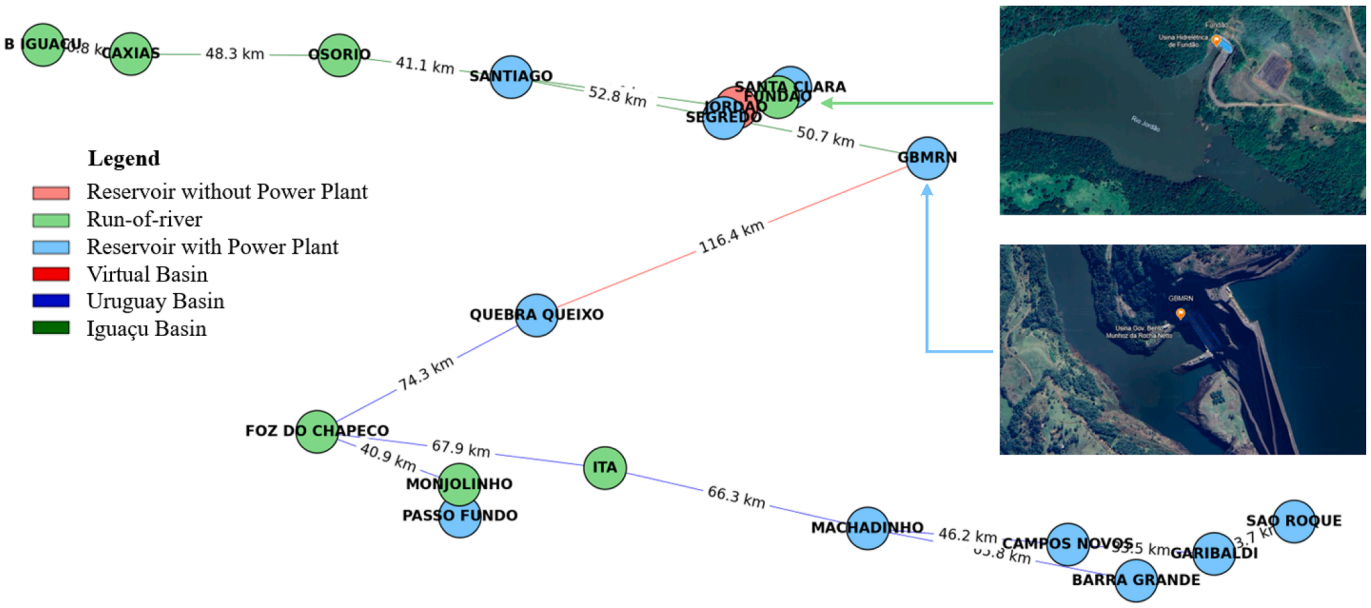

Fig. 3. Graph representation of the interconnected hydroelectric power plants.

to ensure efficient neural network computation. The total processing time encompasses both the training and testing stages of each model. The performance assessment was based on standard regression evaluation metrics, namely the Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE) [37].

3.2. Dataset

This study focused on 19 reservoirs distributed across two river basins. The graph constructed for this analysis captures the hydrological interconnections among these reservoirs. The hydroelectric reservoirs located in the state of Santa Catarina exhibit distinct characteristics influenced by the region's geography and the specific design objectives of each project. Since many of these reservoirs share common river basins, they are naturally interconnected, enabling their representation and analysis through a graph-based modeling approach as proposed in this work. Fig. 3 presents the graph representation of these hydroelectric power plants.

For each reservoir, five operational variables are examined: useful volume percentage, inflow, outflow, energy natural affluent (ENA), and precipitation. The useful volume represents the fraction of the reservoir's total capacity that can be effectively used for electricity generation. The inflow refers to the volume of water entering the reservoir from upstream sources such as rivers, tributaries, or rainfall. The ENA, expressed in average megawatts (MWmed), is computed by the ONS based on the verified inflows and the evolving dynamic configuration of the power system.

The dataset used in this study is sourced directly from the Brazilian National System Operator (ONS), which provides operationally validated and quality-controlled time series for hydroelectric reservoirs. As part of the ONS data curation process, raw measurements undergo consistency checks before public release.

4. Experiments

This section presents the experimental results, beginning with a discussion of the initial data preparation analysis, followed by an ablation study. All experiments use a fixed set of hyperparameters to ensure a controlled comparison across ablation settings. The input and output dimensions are set to four, with an input sequence length of 50 and a prediction horizon of five steps. Models are trained using the Adam optimizer with a learning rate of 10^{-3} and a batch size of 128. The Transformer encoder and decoder each employ a single layer with four attention heads, a feed-forward dimension of 256, and a dropout rate of 0.1. These values were selected to provide sufficient model capacity while keeping the architecture lightweight, allowing the ablation analysis to isolate the impact of individual components rather than effects induced by extensive hyperparameter tuning.

4.1. Preprocessing

During the preprocessing phase, the EWT was applied to detrend the reservoir time series data. This detrending procedure decomposed each time series, including useful volume and inflow, into three components, with the first component representing the low-frequency trend, which was then removed. As illustrated in Fig. 4, the procedure for each reservoir's data can be expressed as:

$$x_{detrended}(t) = x(t) - W_f^{\epsilon}(1,t) \tag{42}$$

where $x(t)$ denotes the original signal, $W_f^{\epsilon}(1,t)$ represents the first EWT component corresponding to the trend, and $x_{\text{detrended}}(t)$ is the resulting detrended signal.

The quantitative analysis presented in Fig. 4 shows that the trend component explains 11.0% of the variance in volume data and 43.9% of the variance in inflow data for the Foz do Chapeco reservoir. This disparity reflects the intrinsic characteristics of the two parameters: inflow displays more pronounced seasonal patterns, whereas volume is influenced by operational management, leading to higher-frequency fluctuations.

The detrending procedure fulfills several key functions within our modeling framework. It standardizes signals across different reservoirs, enhancing model stability. Additionally, it facilitates the identification of correlated fluctuations between reservoirs, as illustrated in Fig. 5, which displays the detrended volume signals for Foz do Chapeco, Machadinho, Barra Grande, and Campos Novos.

By eliminating the trend component while retaining the higher-frequency elements, we extract features that more accurately capture the short-term dynamics between reservoirs, which are crucial for our graph-based modeling approach.

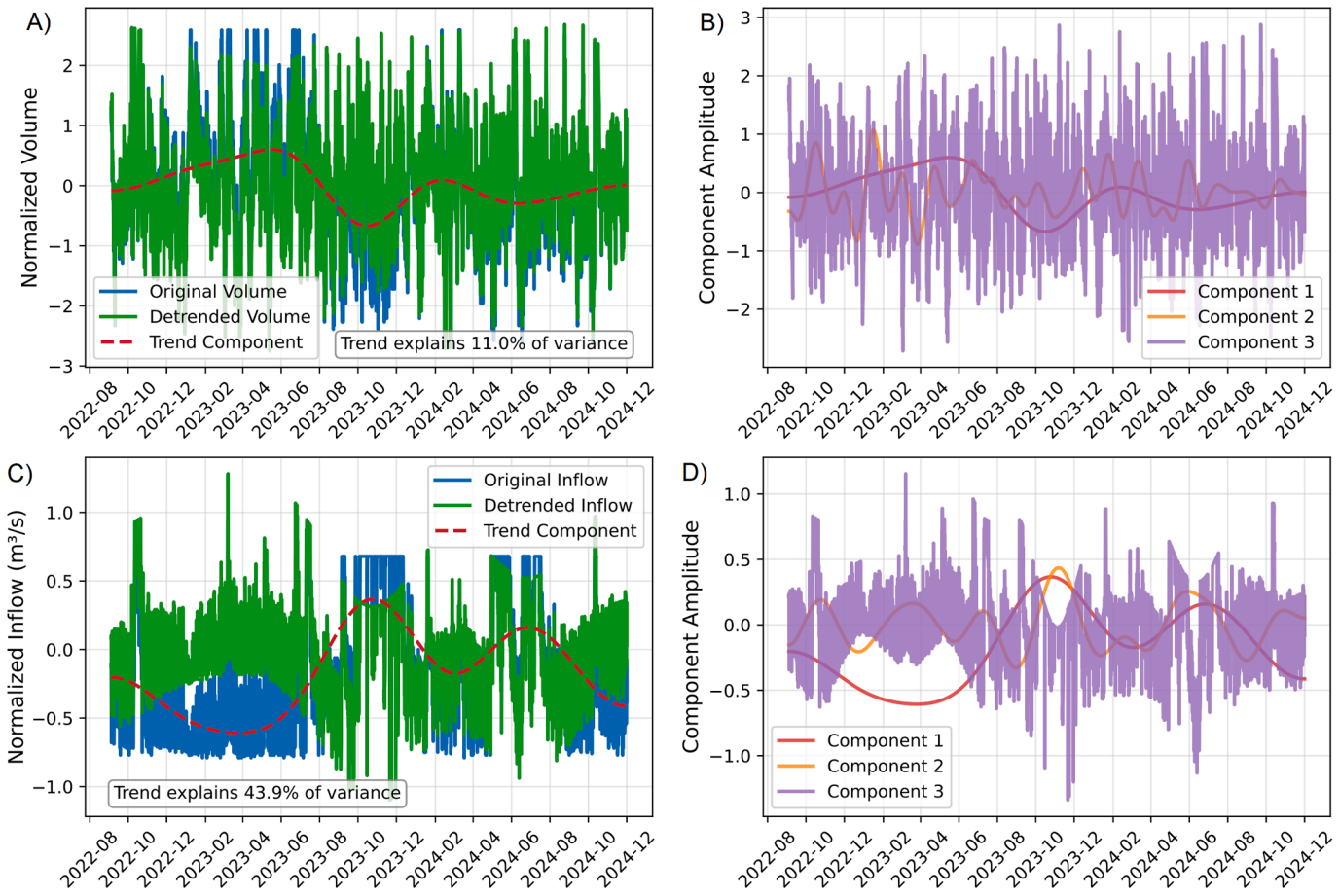

Fig. 4. EWT for Foz do Chapeco reservoir: A) Volume detrending; B) EWT component decomposition of volume data; C) inflow detrending; D) EWT component decomposition of inflow data.

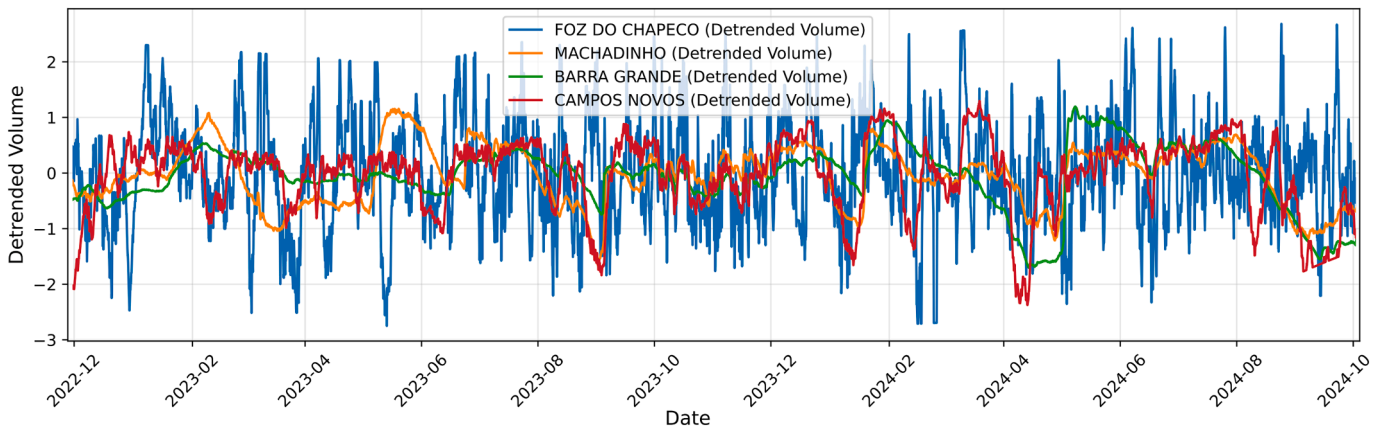

Fig. 5. Comparison of detrended useful volume signals from four reservoirs revealing correlated fluctuation patterns that suggest underlying hydrological interconnections.

4.2. Ablation study

To evaluate the contribution of individual preprocessing heads and the Mixture-of-Experts (MoE) mechanism within the `HeadComposer` module, we conduct a component-wise ablation study. The analysis examines the impact of enabling or disabling RevIN, Decomposition, Multi-Scale Convolution, Patch Embedding, and MoE routing on forecasting performance. All models are trained on a synthetic multivariate time series of 1000 timesteps with four input variables, using a sequence length of 50 and a prediction horizon of five steps. Training is performed for 100 epochs using the Adam optimizer (learning rate 10^{-3}) with mean squared error (MSE) as the loss function.

Table 3 reports the percentage change in each evaluation metric relative to the full model configuration, in which all components are enabled. Positive values indicate performance degradation with respect to this reference.

Disabling all heads leads to a substantial degradation in performance, with MSE increasing by approximately 190%. Among individual components, RevIN has the largest impact: removing it results in increases of 263% in MSE and 146% in MAE, highlighting its role in

Table 3
Component-wise ablation study with accuracy degradation and average inference time. Accuracy metrics are reported as percentage change relative to the full model (all components enabled). Inference time corresponds to the average forward-pass latency per batch.

Components enabled					Performance impact			
RevIN	Decomp.	MS-Conv	Patch	MoE	ΔMSE (%)	ΔRMSE (%)	ΔMAE (%)	Time (ms)
✓	✓	✓	✓	✓	0.00	0.00	0.00	3.72
×	×	×	×	✓	21.18	10.07	10.73	3.43
×	✓	✓	✓	✓	22.02	10.45	10.38	3.65
✓	×	✓	✓	✓	23.20	11.00	11.18	3.69
✓	✓	×	✓	✓	22.41	10.70	10.64	3.59
✓	✓	✓	×	✓	22.40	10.70	10.78	3.68
✓	✓	✓	✓	×	7.27	3.57	3.92	1.91

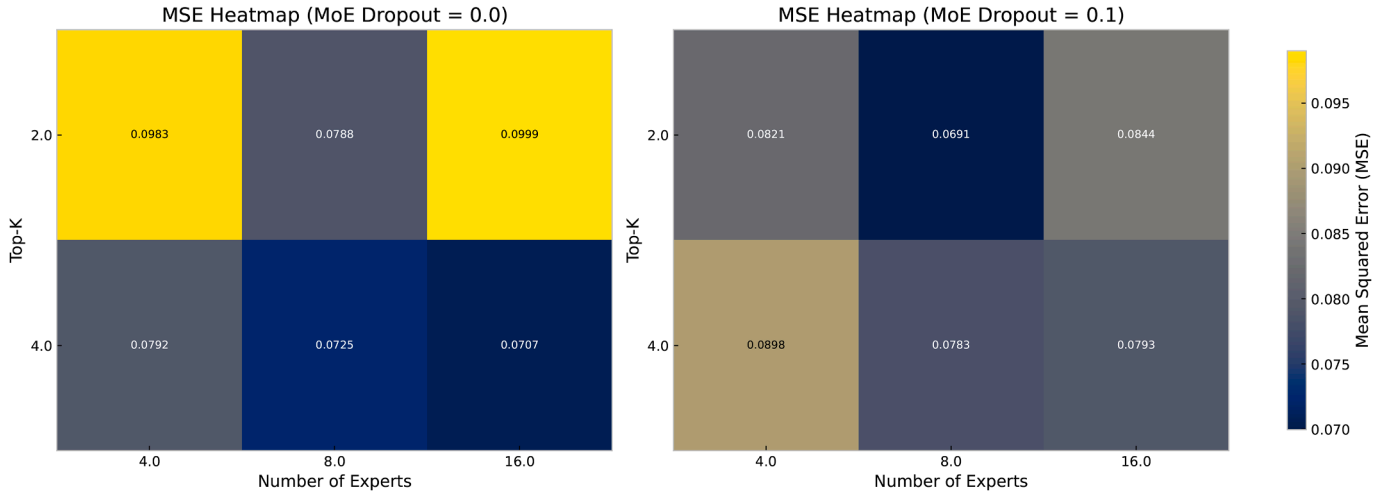

Fig. 6. MoE parameter sensitivity heatmap.

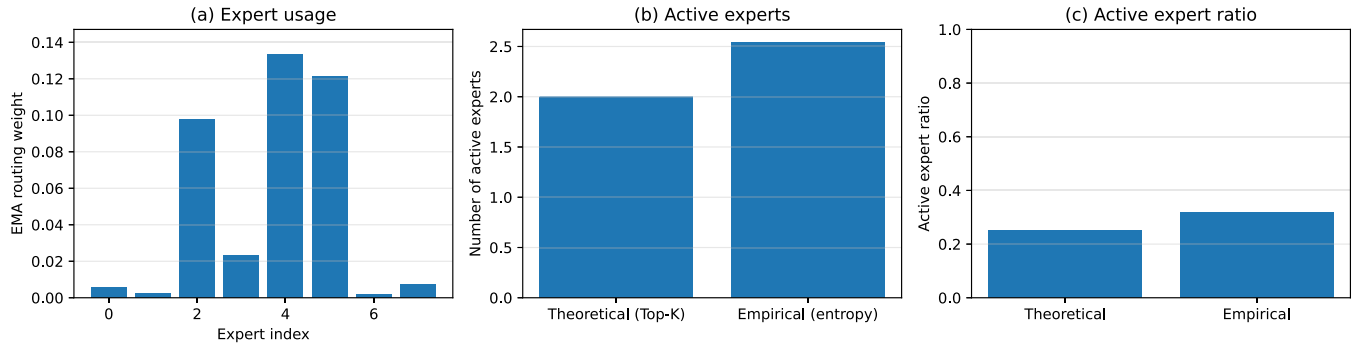

Fig. 7. MoE routing behavior analysis. (a) Average expert usage, computed as an exponential moving average of routing probabilities. (b) Comparison between the theoretical number of active experts implied by Top-K routing and the empirical effective number estimated from routing entropy. (c) Corresponding active expert ratio. Although routing is sparse at the token level, expert utilization remains distributed across the dataset.

mitigating non-stationarity. Patch Embedding also exhibits a noticeable effect, with degradations of approximately 36–37%. In contrast, disabling Decomposition or Multi-Scale Convolution leads to more moderate increases, on the order of 14–15%.

The MoE mechanism exhibits a different behavior. Disabling MoE while keeping all heads active results in relatively smaller degradations (7.8% in MSE, 3.8% in RMSE, and 3.5% in MAE). This indicates that MoE is not the primary driver of performance gains in this configuration. Nevertheless, its contribution is consistent across all metrics, suggesting that expert routing provides complementary improvements by refining representations learned by the preprocessing heads rather than replacing them. These results support the interpretation of MoE as an additive enhancement that improves efficiency and specialization without dominating the overall model behavior.

In addition to accuracy, Table 3 reports the average inference time per forward pass, measured under identical hardware and batch-size conditions. The results show that configurations with MoE incur a higher inference latency compared to the non-MoE counterpart. In particular, disabling MoE while keeping all preprocessing heads active reduces the average inference time from approximately 3.7 ms to 1.9 ms per batch. This confirms that MoE introduces additional computational overhead due to expert routing and increased parameterization. However, this overhead is accompanied by consistent accuracy gains across all evaluated metrics, as reflected by the systematic degradation observed when

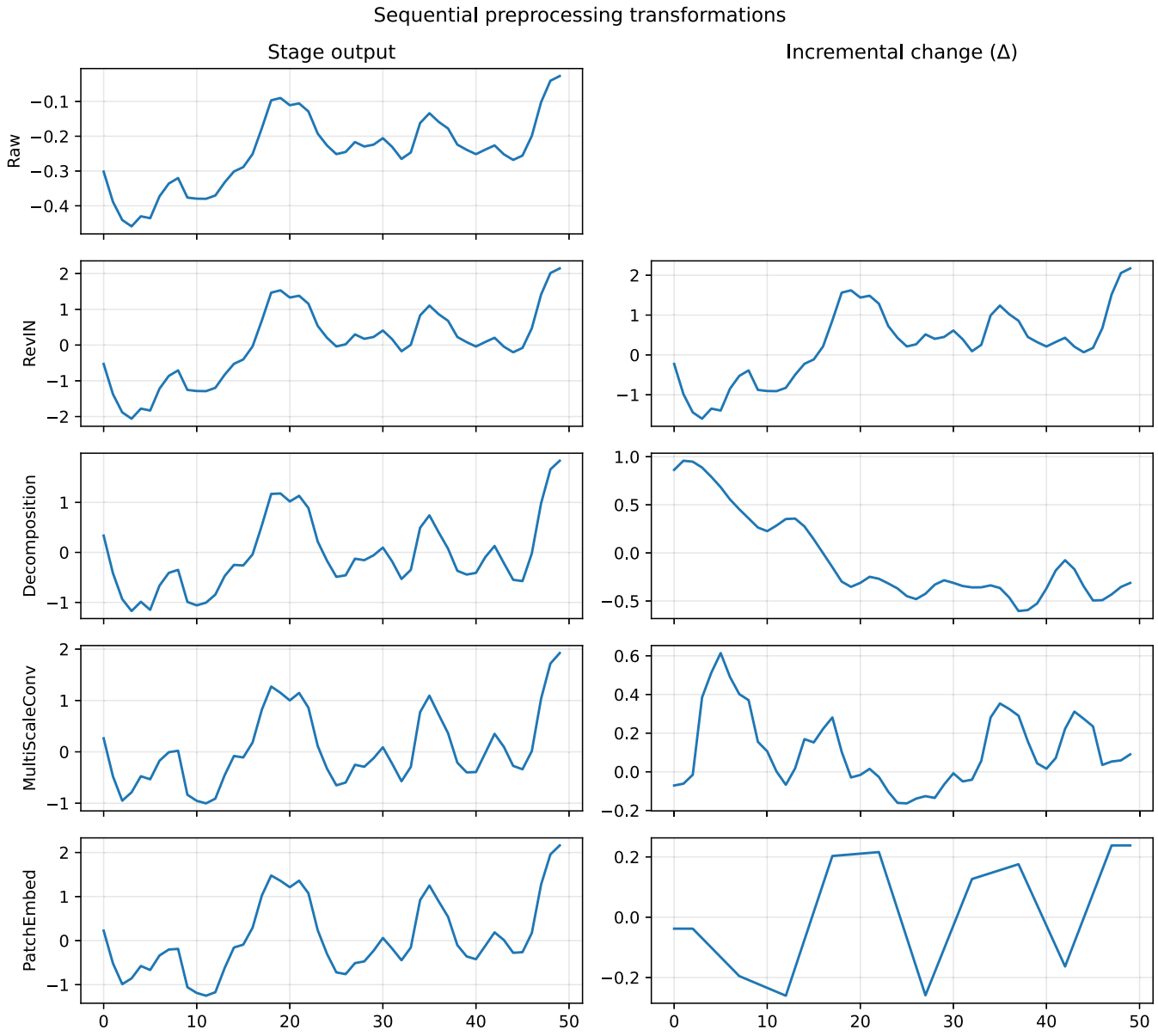

Fig. 8. Sequential preprocessing transformations and incremental changes.

MoE is removed. Therefore, the efficiency of MoE in this setting should be interpreted as an accuracy–complexity trade-off, where improved predictive performance is achieved at the cost of increased inference time rather than reduced computational expense.

4.3. MoE analysis

First, we analyze the sensitivity of the MoE encoder to its routing hyperparameters, namely the number of experts (E), the Top-K routing parameter (K), and the MoE dropout probability (p_{moe}). The analysis is conducted by evaluating multiple configurations obtained from the Cartesian product $E \in \{4, 8, 16\}$, $K \in \{2, 4\}$, and $p_{\text{moe}} \in \{0.0, 0.1\}$. All configurations share the same data splits, optimization procedure, and training budget, isolating the impact of the MoE design choices.

Fig. 6 presents the MSE for each configuration using a heatmap representation. Rows correspond to the Top-K routing parameter and columns to the number of experts, with separate panels for the two MoE dropout settings. A shared color scale is used across panels to facilitate direct comparison.

The results indicate that the relationship between the number of experts and predictive error is not monotonic. For $p_{\text{moe}} = 0.0$, configurations with a larger expert pool ($E = 16$) and low routing sparsity ($K = 2$) exhibit higher MSE values compared to configurations with fewer experts. This suggests that increasing model capacity without additional regularization may negatively affect generalization in this setting.

Introducing MoE dropout ($p_{\text{moe}} = 0.1$) alters the observed performance landscape. Several configurations show reduced MSE relative to their no-dropout counterparts, particularly for intermediate values of E and lower values of K. In contrast, configurations with larger K tend to display smaller performance variations across different expert counts when dropout is applied. Overall, the heatmaps illustrate that MoE performance depends on the joint configuration of expert count, routing sparsity, and dropout, motivating the treatment of routing parameters as coupled design variables rather than tuning them independently.

Routing behavior and expert utilization.. To complement the performance-based sensitivity analysis, we examine the internal

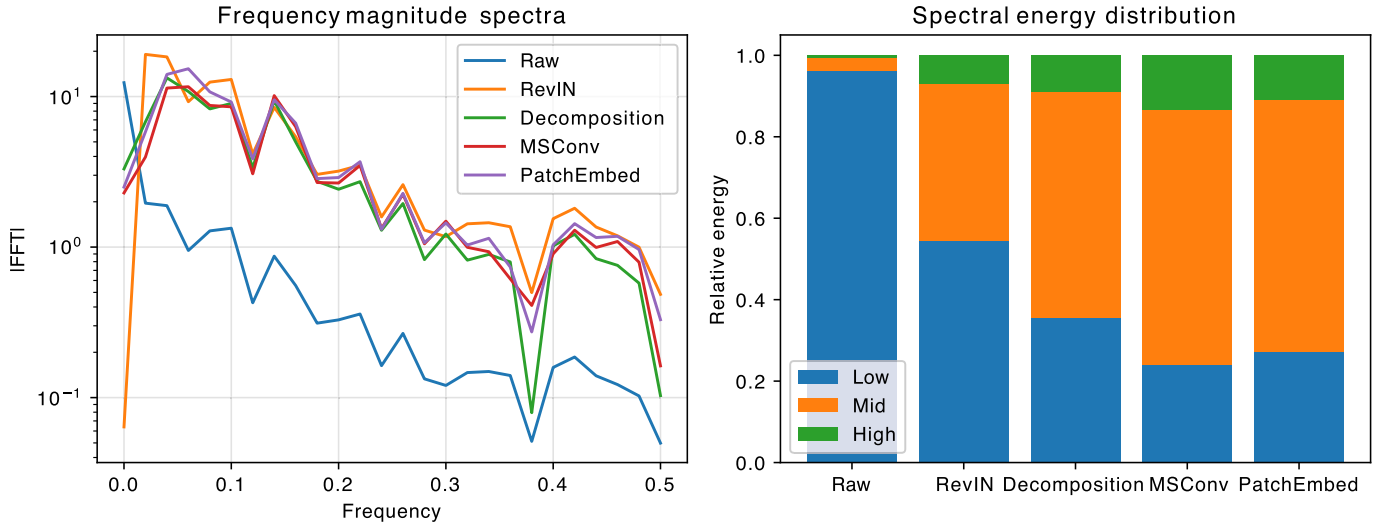

Fig. 9. Frequency-domain analysis and spectral energy redistribution across preprocessing stages.

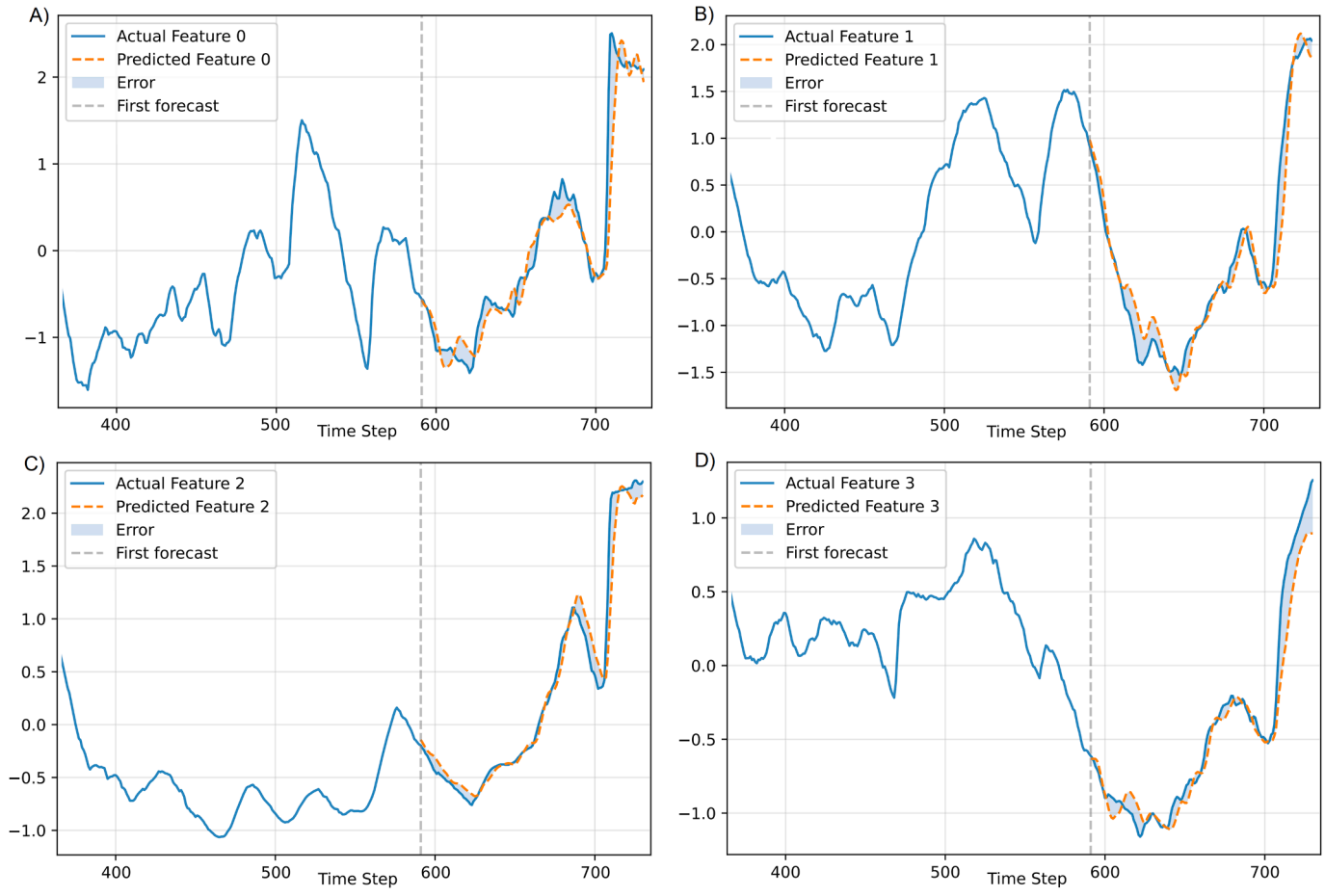

Fig. 10. Sample forecasts for four features: A) feature 0, B) feature 1, C) feature 2, and D) feature 3.

routing behavior of the MoE encoder for representative configurations. Fig. 7 summarizes expert utilization and routing sparsity.

Fig. 7(a) reports the average expert usage, computed as an exponential moving average of routing probabilities aggregated over training iterations. Although Top-K routing enforces a hard per-token constraint, the resulting usage distribution spans multiple experts, indicating that different tokens are routed to different expert subsets rather than repeatedly activating the same experts.

Fig. 7(b) compares the theoretical number of active experts implied by Top-K routing with the empirical effective number of active experts, estimated from the entropy of the routing distribution. While the theoretical minimum is fixed at K, the empirical value is consistently higher, reflecting distributed expert utilization at the dataset level rather than routing collapse.

Finally, Fig. 7(c) reports the corresponding active expert ratio, normalized by the total number of experts. The observed gap between theo-

retical and empirical ratios highlights that sparse per-token computation coexists with global expert diversity, consistent with the intended design of sparse Mixture-of-Experts architectures.

4.4. Analysis of head preprocessing representations

Fig. 8 illustrates the progressive transformation of the input time series by the multi-head preprocessing module. The first column shows the signal after each preprocessing stage, while the second column highlights the incremental change relative to the preceding stage. RevIN primarily performs an affine normalization, stabilizing scale variability while preserving temporal structure. The decomposition head removes low-frequency components associated with long-term trends, yielding a more stationary representation. MultiScaleConv amplifies mid- and high-frequency temporal patterns across multiple receptive fields, introducing localized oscillatory structure. Finally, PatchEmbed compresses the temporal representation, smoothing high-frequency variations while preserving dominant dynamics.

Fig. 9 provides a frequency-domain perspective of these transformations. The magnitude spectra reveal a progressive redistribution of energy from low-frequency components toward multi-resolution and compressed representations. The stacked bar plot further quantifies this effect by decomposing spectral energy into low-, mid-, and high-frequency bands. While the raw and RevIN-normalized signals are dominated by low-frequency content, MultiScaleConv increases mid- and high-frequency energy, and PatchEmbed attenuates high-frequency components due to temporal aggregation.

4.5. Model comparison

To contextualize the performance of our proposed model (referred to as “Ours”), we compare it against a selection of state-of-the-art time series forecasting models. These baselines include transformer-based architectures (e.g., TFT, PatchTST, Autoformer, Informer), MLP-based models (e.g., TiDE, NBEATS), and recurrent models (e.g., LSTM, GRU, DeepAR). The comparison is conducted on the paper dataset, with the same training and evaluation setup: 70% train/30% validation split, sequence length of 50, target length of 5, and 100 training epochs using Adam optimizer (learning rate 0.001) and MSE loss.

Table 4 summarizes the results across key metrics: RMSE, MSE, MAE, and MAPE. Lower values indicate better performance for all metrics. For our model, MAPE is not reported due to the focus on absolute errors in this setting.

The results demonstrate that our model outperforms all baselines in terms of RMSE, MSE, and MAE, achieving approximately 6% improvement in RMSE over the next-best model (TFT) and up to 80% reduction compared to weaker performers like Autoformer. This superiority is attributed to the synergistic integration of diverse heads via `HeadComposer`, which enhances feature representation without substantially increasing complexity, as evidenced by the Informer baseline’s higher errors despite a similar architecture.

Among baselines, transformer variants (e.g., TFT, PatchTST) generally outperform recurrent and MLP models, consistent with prior findings [27]. However, our approach further bridges gaps in multivariate handling and long-range dependencies. Overall, these comparisons validate `HeadComposer` as a versatile enhancement for improving forecasting accuracy across architectures.

Fig. 10 shows the resulting prediction for four of the studied features. For all of the analyzed features, our model presented predicted results that were close to the observed values, proving that the proposed model is indeed promising for the given task.

Table 4
Model performance comparison on the paper dataset. Lower is better for all metrics. Our model achieves the lowest RMSE, MSE, and MAE.

Model	RMSE	MSE	MAE	MAPE
Ours	0.249	0.062	0.145	–
TFT	0.265	0.070	0.234	26.436
PatchTST	0.312	0.097	0.256	32.260
TimeMixer	0.337	0.113	0.296	38.750
TimesNet	0.341	0.116	0.322	42.786
NHITS	0.353	0.125	0.302	35.278
LSTM	0.427	0.182	0.367	60.596
NBEATS	0.427	0.183	0.338	31.096
NBEATSx	0.427	0.183	0.338	31.096
GRU	0.457	0.209	0.376	59.830
DeepAR	0.500	0.250	0.410	49.402
TCN	0.578	0.334	0.434	78.288
BiTCN	0.602	0.362	0.490	65.313
iTransformer	0.612	0.374	0.527	97.922
TimeXer	0.679	0.461	0.535	66.640
Informer	0.726	0.527	0.598	51.648
VanillaTransformer	0.810	0.656	0.706	91.263
StemGNN	0.823	0.677	0.651	55.278
TiDE	0.941	0.885	0.728	82.724
Autoformer	1.230	1.513	1.077	128.798

4.6. Limitations

Considering that reservoir forecasts are based on trends from other reservoirs (considering a graph structure), when significant variations occur in a reservoir, the model may not predict this variation since the forecast is based on the variation of the entire system (within the same hydrographic basin). This result may occur due to reservoir management decisions specific to the plant, increasing the forecast error for this reservoir.

While the proposed hybrid forecasting framework demonstrates strong performance, several key limitations warrant consideration. Firstly, the model’s computational demands are substantial, requiring high-end hardware, which may restrict accessibility and scalability for real-time or resource-constrained deployments. Secondly, the evaluation is confined to a geographical context with several reservoirs, which may limit the generalizability of the findings to other hydrological systems with different climatic, operational, or topological characteristics. Finally, the study focuses on short-term predictions, leaving open evaluations about the framework’s effectiveness for longer forecasting horizons.

5. Conclusion

This study presented a hybrid forecasting framework that integrates advanced preprocessing techniques with a sparse MoE–enhanced Transformer architecture for short-term hydroelectric reservoir volume prediction. By combining the Empirical Wavelet Transform for denoising with multi-head feature extraction mechanisms, comprising reversible instance normalization, seasonal–trend decomposition, multi-scale convolution, and patch embedding, the model effectively captures both local and global temporal dependencies inherent in hydroelectric systems.

The accurate prediction of hydroelectric reservoir volumes, as demonstrated in the study, plays a critical role in mitigating market price volatility and operational uncertainties in electricity markets. By providing more reliable short-term forecasts of water availability and generation potential, the proposed model enables system operators and market participants to optimize bidding strategies, reduce reliance on costly backup generation, and better balance supply and demand.

Enhanced forecasting accuracy helps diminish the uncertainties associated with hydrological variability, such as unexpected inflow fluctuations or seasonal shifts, which often lead to price spikes and supply instability. Consequently, improving reservoir volume prediction not only supports more stable and efficient market operations but also contributes to greater financial predictability and reduced risk in hydro-dominated power systems.

Experimental evaluations conducted on 19 reservoirs across the Uruguay and Iguaçu basins demonstrated that the proposed model significantly outperforms 18 state-of-the-art deep learning baselines, achieving the lowest RMSE, MSE, and MAE values. The inclusion of conditional computation through sparse expert routing improved model capacity without compromising computational efficiency, making the framework suitable for large-scale operational deployment.

The results highlight that incorporating diverse preprocessing heads enhances the robustness and generalization of Transformer-based architectures for hydrological forecasting tasks. Moreover, the empirical findings confirm the importance of denoising and trend decomposition in improving short-term predictive accuracy in interconnected reservoir systems.

Future research will focus on extending the model to multi-horizon forecasting with dynamic expert selection, integrating graph-based spatial dependencies directly within the Transformer layers, and exploring transfer learning approaches to improve adaptability across distinct hydrological regions. Overall, the proposed framework establishes a promising direction for accurate and efficient forecasting in sustainable hydroelectric power management.

CRediT authorship contribution statement

Laio Oriel Seman: Writing – original draft, Software; **Kin-Choong Yow:** Writing – review & editing, Supervision; **Stefano Frizzo Stefenon:** Writing – original draft, Methodology, Investigation.

Data availability

Data will be made available on request.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC), funding reference number DDG-2024-00035. Cette recherche a été financée par le Conseil de recherches en sciences naturelles et en génie du Canada (CRSNG), numéro de référence DDG-2024-00035. We thank the Brazilian National Council for Scientific and Technological Development (CNPq).

References

[1] J.M. Damazio, M.A. dos Santos, The multiple services of hydropower plants in the context of sustainable electric sector in Brazil, Energy Explor. Exploit. 42 (3) (2024) 1131–1144.
[2] J.D.P. Mendieta, I.G. Hidalgo, F. Cioffi, Impact of different hydrological models on hydroelectric operation planning, Renew. Energy 232 (2024) 120975.
[3] R.N. Muniz, S.F. Stefenon, W.G. Buratto, A. Nied, R. Cardoso, C.K. Yamaguchi, K.-C. Yow, Time series forecasting based on multi-criteria optimization for model and filter selection applied to hydroelectric power plants, Energy 337 (2025) 138688.
[4] S. Di Grande, M. Berlotti, S. Cavalieri, R. Gueli, A machine learning approach to forecasting hydropower generation, Energies 17 (20) (2024) 5163.
[5] S.F. Stefenon, L.O. Seman, C.K. Yamaguchi, L.D.S. Coelho, V.C. Mariani, J.P. Matos-Carvalho, V.R.Q. Leithardt, Neural hierarchical interpolation time series (NHITS) for reservoir level multi-horizon forecasting in hydroelectric power plants, IEEE Access 13 (2025) 54853–54865.
[6] R.A. Mosqueira Furtado, A.L. Marques Marcato, I.C. da Silva Junior, Generation of synthetic series for hydroelectric plants in Brazilian interconnected system by metaheuristic, in: International Universities Power Engineering Conference (UPEC), 59, IEEE, Cardiff, United Kingdom, 2024, pp. 1–6.
[7] H. Liu, J. Zhao, C. Pan, L. Mao, Data-driven real-time wind power forecasting based on the dynamic adaptive selective state space model (DA-SSSM), Electr. Power Syst. Res. 250 (2026) 112176.
[8] A.B.A. Ferreira, J.B. Leite, D.H.P. Salvadeo, Power substation load forecasting using interpretable transformer-based temporal fusion neural networks, Electr. Power Syst. Res. 238 (2025) 111169.
[9] P. Lin, X. Zhang, L. Gong, J. Lin, J. Zhang, S. Cheng, Multi-timescale short-term urban water demand forecasting based on an improved PatchTST model, J. Hydrol. 651 (2025) 132599.
[10] M. Gong, Z. Wu, F. Zhang, M. Sun, J. Huang, Y. Tian, W. Xu, Adaptive multi-scale short term wind power forecasting framework based on TimeMixer and mix-of-experts, Electr. Power Syst. Res. 251 (2026) 112196.
[11] J. Gong, Z. Qu, Z. Zhu, H. Xu, Parallel TimesNet-BiLSTM model for ultra-short-term photovoltaic power forecasting using STL decomposition and auto-tuning, Energy 320 (2025) 135286.
[12] H. Sun, X. Fu, Y. Luo, Z. Wu, C. Peng, A multistep wind power prediction model based on sample entropy clustering decomposition and inverted transformer, Electr. Power Syst. Res. 251 (2026) 112286.
[13] Q. Liu, J. Cao, J. Zhang, Y. Zhong, T. Ba, Y. Zhang, Short-term power load forecasting in FGSM-Bi-LSTM networks based on empirical wavelet transform, IEEE Access 11 (2023) 105057–105068.
[14] B.L. Tuleski, C.K. Yamaguchi, S.F. Stefenon, L. dos Santos Coelho, V.C. Mariani, Audio-based engine fault diagnosis with wavelet, markov blanket, ROCKET, and optimized machine learning classifiers, Sensors 24 (22) (2024) 7316.
[15] T. Kim, J. Kim, Y. Tae, C. Park, J.-H. Choi, J. Choo, Reversible instance normalization for accurate time-series forecasting against distribution shift, in: International Conference on Learning Representations, 2022.
[16] Y. Nie, N.H. Nguyen, P. Sinthong, J. Kalagnanam, A time series is worth 64 words: long-term forecasting with transformers, in: International Conference on Learning Representations, 2023.
[17] X. Shi, S. Wang, Y. Nie, D. Li, Z. Ye, Q. Wen, M. Jin, Time-MoE: billion-scale time series foundation models with mixture of experts, arXiv:2409.16040 4 (2024) 1–33.
[18] X. Liu, J. Liu, G. Woo, T. Aksu, Y. Liang, R. Zimmermann, C. Liu, S. Savarese, C. Xiong, D. Sahoo, Moirai-MoE: empowering time series foundation models with sparse mixture of experts, arXiv:2410.10469 1 (2024) 1–20.
[19] H. Liu, K. Simonyan, Y. Yang, Darts: differentiable architecture search, arXiv:1806.09055 2 (2018) 1–13.
[20] K.G. Olivares, C. Challu, F. Garza, M.M. Canseco, A. Dubrawski, NeuralForecast: user friendly state-of-the-art neural forecasting models, 2022, (PyCon Salt Lake City, Utah, USA).
[21] M. Chen, H. Peng, J. Fu, H. Ling, Autoformer: searching transformers for visual recognition, in: International Conference on Computer Vision, IEEE/CVF, 2021, pp. 12270–12280.
[22] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, R. Jin, Fedformer: frequency enhanced decomposed transformer for long-term series forecasting, in: International Conference on Machine Learning, PMLR, 2022, pp. 27268–27286.
[23] Y. Nie, N.H. Nguyen, P. Sinthong, J. Kalagnanam, A time series is worth 64 words: long-term forecasting with transformers, in: The Eleventh International Conference on Learning Representations, ICLR, 2022, pp. 1–24.
[24] J. Zhang, Q. Zhang, G. Li, J. Wu, C. Wang, Z. Li, Hybrid model for renewable energy and load forecasting based on data mining and EWT, J. Electr. Eng. Technol. 17 (3) (2022) 1517–1532.
[25] A. Zeng, M. Chen, L. Zhang, Q. Xu, Are transformers effective for time series forecasting?, in: Proceedings of the AAAI Conference on Artificial Intelligence, 37, 2023, pp. 11121–11128.
[26] Y. Liu, T. Hu, H. Zhang, H. Wu, S. Wang, L. Ma, M. Long, iTransformer: inverted transformers are effective for time series forecasting, arXiv:2310.06625 (2024).
[27] B. Lim, S.Ö. Arık, N. Loeff, T. Pfister, Temporal fusion transformers for interpretable multi-horizon time series forecasting, Int. J. Forecast. 37 (4) (2021) 1748–1764.
[28] Y. Zhang, L. Ma, S. Pal, Y. Zhang, M. Coates, Multi-resolution time-series transformer for long-term forecasting, in: International Conference on Artificial Intelligence and Statistics, PMLR, 2024, pp. 4222–4230.
[29] W. Xue, T. Zhou, Q. Wen, J. Gao, B. Ding, R. Jin, CARD: channel aligned robust blend transformer for time series forecasting, arXiv:2305.12095 5 (2024) 1–39.
[30] H. Wu, J. Xu, J. Wang, M. Long, Autoformer: decomposition transformers with auto-correlation for long-term series forecasting, in: Advances in Neural Information Processing Systems, 34, 2021, pp. 22419–22430.
[31] A. Bhayana, I. Shpilfoygel, S. Sun, M. Rege, Decompose and conquer: time series forecasting with multiseasonal trend decomposition using loess, Forecasting 5 (4) (2023) 684–696.
[32] E.C. da Silva, E.C. Finardi, S.F. Stefenon, Enhancing hydroelectric inflow prediction in the Brazilian power system: a comparative analysis of machine learning models and hyperparameter optimization for decision support, Electr. Power Syst. Res. 230 (2024) 110275.
[33] S.F. Stefenon, L.O. Seman, E.C. da Silva, E.C. Finardi, C.L. dos Santos, V.C. Mariani, Hypertuned wavelet convolutional neural network with long short-term memory for time series forecasting in hydroelectric power plants, Energy 313 (2024) 133918.
[34] L. Lyu, M. Fang, N. Wang, J. Wu, Water level prediction model based on GCN and LSTM, in: 2021 7th International Conference on Computer and Communications (ICCC), IEEE, 2021, pp. 1600–1605.
[35] J. Huan, W. Liao, Y. Zheng, X. Xu, H. Zhang, B. Shi, A deep learning model with spatio-temporal graph convolutional networks for river water quality prediction, Water Supply 23 (7) (2023) 2940–2957.
[36] Z. Chen, J. Wang, M. Wei, Research on power generation prediction of hydropower in river basin based on multi-head attention graph convolutional neural network, J. Comput. Methods Sci. Eng. 24 (2) (2024) 797–811.
[37] S.F. Stefenon, K.-C. Yow, Adaptive filter-driven optimized attention-based CNN-LSTM for load forecasting in microgrids, Results Eng. 28 (2025) 107470.