applied sciences

MDPI

Article

Post Constraint and Correction: A Plug-and-Play Module for Boosting the Performance of Deep Learning Based Weather Multivariate Time Series Forecasting

Zhengrui Wang, Zhongwen Luo *, Zirui Yang and Yuanyuan Liu

School of Computer Science, China University of Geosciences (Wuhan), Wuhan 430078, China; 20181003704@cug.edu.cn (Z.W.); yzr_cug_25@163.com (Z.Y.); liuyy@cug.edu.cn (Y.L.)
* Correspondence: luozw@cug.edu.cn

Abstract: Weather forecasting is essential for various applications such as agriculture and transportation, and relies heavily on meteorological sequential data such as multivariate time series collected from weather stations. Traditional numerical weather prediction (NWP) methods applied to multivariate time series forecasting are grounded in statistical principles such as Autoregressive Integrated Moving Average (ARIMA); however, they often struggle with capturing complex nonlinear patterns among meteorological variables and temporal variances. Currently, existing deep learning approaches such as Recurrent Neural Networks (RNNs) and transformers offer remarkable performance in handling complex patterns among meteorological multivariate time series, yet frequently fail to maintain weather-specific physical properties such as strict values constraints, while also incurring the significant computational costs of large parameter scales. In this paper, we present a novel deep learning plug-and-play framework named Post Constraint and Correction (PCC) to address these challenges by incorporating additional constraints and corrections based on weather-specific properties such as multivariant correlations and physical-based strict value constraints into the prediction process. Our method demonstrates notable computational efficiency, delivering significant improvements over existing deep learning time series models and helping to achieve better performance with far fewer parameters. Extensive experiments demonstrate the effectiveness, efficiency, and robustness of our method, highlighting its potential for real-world applications.

Keywords: deep learning; multivariate time series forecasting; weather multivariate time series

check for updates

Academic Editor: Grigorios Beligiannis

Received: 23 January 2025
Revised: 15 March 2025
Accepted: 27 March 2025
Published: 3 April 2025

Citation: Wang, Z.; Luo, Z.; Yang, Z.; Liu, Y. Post Constraint and Correction: A Plug-and-Play Module for Boosting the Performance of Deep Learning Based Weather Multivariate Time Series Forecasting. *Appl. Sci.* **2025**, *15*, 3935. https://doi.org/10.3390/app15073935

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Weather forecasting plays a crucial role in modern society, impacting various sectors from agriculture and transportation to daily life planning [1]. Meteorological multivariate time series collected from weather stations serve as one of the primary data sources for weather forecasting tasks, containing multiple variables such as temperature, humidity, pressure, wind velocity, and precipitation which can be used to describe the weather state at a specific location over a period of time.

Traditional numerical weather prediction (NWP) methods applied to multivariate time series forecasting tasks primarily consist of statistical and mathematical models, including Autoregressive Integrated Moving Average (ARIMA) [2], Kalman filtering [3], and Singular Spectrum Analysis (SSA) [4]. However, these which often struggle with nonlinear relationships and complex patterns among meteorological series [5]. Recently, the advancement of deep learning has transformed weather forecasting capabilities, driven by three key factors:

algorithmic innovations, the unprecedented scale of available weather data, and advances in parallel computing hardware [6]. Deep learning time series forecasting models excel at automatically extracting complex temporal patterns and dependencies from time series data, achieving state-of-the-art accuracy [7,8].

Meteorological variables possess distinct physical characteristics that significantly influence their temporal dynamics and interrelationships. Temperature exhibits daily and seasonal cyclical patterns with gradual transitions, while precipitation demonstrates intermittent and often non-Gaussian distributions with sudden onsets and varying intensities [9]. Atmospheric pressure typically displays smoother temporal transitions, but can undergo abrupt changes during weather front passages. These variables are further governed by fundamental physical principles such as energy conservation and thermodynamic laws that impose constraints on their possible values and rates of change [10].

However, deep learning models often fail to properly account for meteorology-specific patterns of different meteorological variables such as the strict physical constraints and boundary conditions inherent in weather systems, potentially leading to imprecise or unreliable predictions. Meanwhile, due to the complexity and dynamic nature of these physical characteristics, which exhibit subtle or significant variations across different time periods and geographical regions [11], predefining these relationships in a static way would inevitably affect prediction accuracy and generalization capability. Furthermore, many advanced deep learning models employ complex architectures with massive parameter counts, resulting in high computational resource requirements. These issues limit their practical deployment in real-world timely weather forecasting systems where both accuracy and computational efficiency are critical [12].

To address these challenges, we propose a novel deep learning plug-and-play framework named Post Constraint and Correction (PCC). PCC explicitly incorporates data characteristics that emerge from complex physical processes into the normal prediction process, including numerical constraints and variable relationships, thereby contributing to prediction reliability. Our approach introduces two key sub-modules: (1) the Multi-variants Correlation Constraint (MCC) module, which captures and maintains the complex interdependencies between different meteorological variables, and (2) the State Correction (SC) module, which ensures the reasonability of predictions by incorporating additional correction terms into the final prediction. By introducing network modules that can adaptively learn the empirical relationships between meteorological variables, deep learning models can effectively capture the underlying dynamics without requiring precise definitions of basic laws.

Notably, our method adopts a linear architectural design that achieves significantly efficient computational complexity and parameter scale while delivering notable accuracy improvements, making it suitable for practical weather forecasting. The code implementation is available at a GitHub repository: https://github.com/Fubukipara/PCCforWeatherTS/ (accessed on 26 March 2025).

The main contributions of our work are threefold:

1. We develop a plug-and-play deep learning module named PCC that incorporates variable relationships and maintains state reasonability for weather multivariate time series forecasting, enabling seamless integration with various forecasting models without requiring architectural modifications or additional preprocessing.
2. We design a computationally efficient architecture that significantly improves backbone model performance on weather time series forecasting tasks with minimal additional computational overhead, allowing the enhanced model to achieve superior performance with a significantly reduced parameter count.

3. We conduct comprehensive experiments, ablation studies, and visualizations to demonstrate and analyze the superiority of our method.

2. Related Work

2.1. Weather Multivariate Time Series Forecasting

Multivariate time series data collected from weather stations serve as a fundamental cornerstone in modern weather forecasting systems. These data are characterized by their high dimensionality and complex temporal dependencies [13]. Traditional weather forecasting methods based on multivariate time series data primarily rely on various statistical approaches, including Autoregressive Integrated Moving Average (ARIMA) models, which assume linear relationships between variables, along with extensions such as SARIMA [14] for seasonal data. Other techniques have also been widely used, such as exponential smoothing [15] and Kalman filtering. Although these methods are interpretable and easy to implement, they often struggle to capture the complex and nonlinear relationships inherent in meteorological data, which may lead to unreliable predictions.

Driven by improvements in deep learning algorithms along with increases in the scale of available weather data and parallel computational hardware resources, recent advances in deep learning have revolutionized the field of weather forecasting and enabled the training of larger-scale models [16]. These new models are capable of effectively capturing complex and nonlinear patterns and dependencies within meteorological data in an end-to-end manner while eliminating the need for time-consuming manual feature engineering [17]. Thus, interest in deep learning models for NWP tasks there has been developing rapidly. For example, multiple deep learning methods such as Pangu-Weather [18] and FengWu [19] based on global-scale meteorological data in mesh space have shown the ability to outperform traditional physical models in terms of both prediction performance and efficiency.

For multivariate time series point data collected from weather stations, Recurrent Neural Networks (RNNs) and variants such as Gated Recurrent Unit (GRU) and Long Short-Term Memory (LSTM) networks have demonstrated remarkable success in prediction and analyzing tasks [20]. The stacked LSTM (stackLSTM) architecture stands out for its exceptional capacity to capture complex temporal dynamics at various time scales in weather time series by leveraging multiple LSTM layers [21–23]. Recently, several studies have explored the potential of transformers in the context of weather multivariate time series [24], leveraging the strength of transformers in capturing complex or long-term dependencies in sequential data to offer a promising alternative to traditional RNN-based approaches.

2.2. Deep Learning-Based Time Series Forecasting

Time series forecasting encompasses a broad spectrum of applications, ranging from financial markets [25] and traffic [26] to weather prediction. While traditional statistical methods such as ARIMA and VAR remain valuable for their interpretability, deep learning approaches have emerged as the dominant paradigm, offering superior performance in handling complex temporal and spatial patterns [27].

The evolution of deep learning models for time series analysis has been marked by significant architectural innovations. RNNs have been the primary choice for handling sequential data for the past several years, including NLP [28] and time series analysis [29]. Improved RNN architectures such as GRU [30] and LSTM [31] have addressed the fundamental limitations of vanilla RNNs in capturing long-range dependencies. These architectures have demonstrated remarkable successes [32], including in weather-related time series applications. However, there exists a significant computational time assumption due

to the autoregressive structure of RNNs, and they also suffer from the gradient vanishing problem. To address these issues, SegRNN introduces parallel prediction mechanisms [33], enabling the model to predict multiple time steps in parallel. This approach improves both forecasting accuracy and computational efficiency.

Nowadays, the transformer architecture has gained significant attention in long sequence modeling [34] thanks to its multi-head self-attention mechanism, which has demonstrated remarkable ability to capture long-term dependencies and avoid the gradient vanishing problem that arises in RNNs. Consequently, transformers have shown great promise in sequential tasks such as time series forecasting. Although the attention mechanism demonstrates superior performance in long-term temporal modeling, it has drawbacks in terms of model complexity and computational cost [35]. Indeed, numerous methods have been proposed to address these issues, such as treating split patches or whole sequences as tokens in PatchTST [36] and iTransformer [37]. Another research line focuses on utilizing intricate temporal patterns within time series by leveraging techniques such as seasonal trend decomposition [38] and nonstationary compensation [39] to improve forecasting accuracy.

Recent developments in time series forecasting have also explored alternative architectures. MLP-based models such as DLinear [40] offer simple yet effective solutions for time series prediction, achieving comparable performance while requiring significantly fewer computational resources.

However, the existing deep learning-based time series forecasting models still lack the ability to capture the specific physical features of weather time series, including the strict values constraints of meteorological variables and their interdependencies, which may lead to suboptimal performance on weather time series prediction tasks. Therefore, in this paper we propose a generic deep learning plug-and-play module to help deep learning-based time series forecasting models to address these challenges effectively and efficiently.

3. Method

To deal with the specific features of weather time series which make deep learning-based time series forecasting models less effective, we propose a novel deep learning plug-and-play module called Post Constraint and Correction (PCC). The PCC module is designed to help the forecasting model capture more appropriate features of weather time series, resulting in more reliable predictions. As we demonstrate in Figure 1, the PCC module consists of two parts: a Multi-variants Correlation Constraint (MCC) module and a State Correction (SC) module. We introduce each part in detail in the following sections. We define the observed weather time series as $X \in \mathbb{R}^{O \times N}$, where O is the length of observed series and N is the number of variants. The ground truth and future prediction are denoted as $\overline{Y} \in \mathbb{R}^{P \times N}$ and $Y \in \mathbb{R}^{P \times N}$, respectively, where P is the prediction horizon (the number of future time steps predicted by the model).

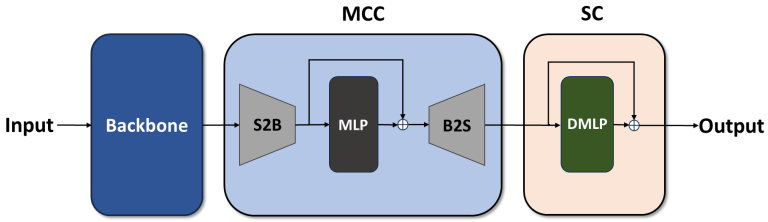


Figure 1. The architecture of our PCC module, which consists of two main submodules: Multi-variants Correlation Constraint (**MCC**) and State Correction (**SC**). The **Backbone** denotes the original time series forecasting model that we aim to enhance, while **S2B** indicates conversion of the time series to the bias matrix and **B2S** denotes conversion of the bias matrix back to a time series. **MLP** denotes the Multi-Layer Perceptron, including the fully connected layer, activation function, and dropout layer, while **DMLP** denotes the MLP with differential representation.

3.1. Initial Prediction

We first feed the observed weather time series X to the backbone model (the underlying time series forecasting model, such as LSTM, that serves as the foundation for forecasting). Then, the backbone model generates the initial prediction Y_i:

$$Y_i = Backbone(X) \tag{1}$$

where $Y_i \in \mathbb{R}^{P \times N}$ is the initial prediction of the future series. The backbone model can be any general time series forecasting model, such as an RNN, transformer, or MLP model. Our PCC module is designed to be plug-and-play, which means that there is no need to change backbone model's architecture, data normalization, training trajectory, etc. It is only necessary to deploy the PCC module on the forecasting series' channel dimension after the backbone model.

3.2. Multi-Variants Correlation Constraint

There is a distinct correlation among different meteorological variables (e.g., temperature, precipitation) that play important roles in climate change [41]. Thus, we consider that offering additional insights on these relationships may help the model to make more accurate predictions.

However, unlike some other multivariate time series data, such as traffic flow and electricity demand, the correlation patterns among meteorological variables are primarily or directly based on physical properties [42] which are strict, specific, stable, and hard to modify or change across the time. However, some methods focus on whole sequences or patches of different channels. For instance, iTransformer considers every channel's sequence as a token, which may introduce irrelevant temporal variances and lead to less effective capture of these temporally independent physical correlation patterns.

To address the above issues, we design the Multi-variants Correlation Constraint (MCC) module, which is deployed after the backbone model on the channel dimension of the forecasting series. The detailed architecture of the MCC's network is shown in Figure 2.

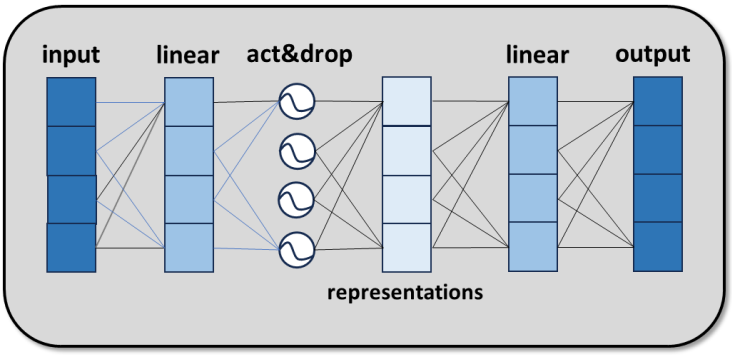


Figure 2. Architecture of the MCC constraint module deployed in an MLP.

After the backbone generates the initial prediction Y_i, we first define the last observed time step X_O as the reference point and representation of the historical states, then calculate the bias between the initial prediction Y_i and X_O:

$$B = Y_i - X_O \tag{2}$$

where $B \in \mathbb{R}^{P \times N}$ is the bias matrix. The MCC module leverages the bias matrix B as a physically meaningful representation of how each variable evolves from the reference state. By using the change relative to a reference point (X_O) rather than absolute values, the module is able to effectively capture the temporal dynamics of weather evolution.

Then, we deploy a fully-connected layer f_{mp} and a nonlinear activation function σ on the channel dimension of B to project the bias of every time steps into the hidden space:

$$h_{mc} = \sigma(f_{mp}(B)) \tag{3}$$

where $h_{mc} \in \mathbb{R}^{P \times H}$ is the hidden representation of the variable correlation and H is the predefined dimension of the hidden space. We deploy a dropout layer after the hidden representation h_{mc} to avoid overfitting [43]:

$$h_{mc} = Dropout(h_{mc}). \tag{4}$$

The fully connected layer f_{mp} learns the complex interdependencies between changes in different meteorological variables through its learnable weights. For example, it can model how changes in temperature relate to corresponding changes in humidity, air pressure, and other variables. The nonlinear activation function σ enables modeling of the nonlinear patterns inherent in complex and dynamic meteorological systems.

Then, another fully connected layer f_{mc} is introduced to generate the constraint term C_{mc}:

$$C_{mc} = f_{mc}(h_{mc}). \tag{5}$$

Next, we add the constraint term C_{mc} to the initial prediction's bias B to generate the constrained bias B_{mc}:

$$B_{mc} = B + C_{mc}. \tag{6}$$

Finally, we convert the constrained bias B_{mc} back to a time series by adding the reference point X_O:

$$Y_{mc} = B_{mc} + X_O. \tag{7}$$

During the process above, the variable correlation constraint term C_{mc} is inferred by the MLP through the bias matrix B separately and independently at every time step, thereby avoiding the irrelevant temporal variances and leading to more precise capture of the variable correlation patterns. Unlike fixed correlation rules, the MCC module can dynamically adjust the relationships between meteorological variables through the learnable weights of the linear layers, thereby enhancing the accuracy and reliability of the model's predictions.

During the training stage, the MCC module can effectively learn the mapping between the bias matrix and the constraint term through a loss-based gradient propagation mechanism:

$$\theta_{MCC} \leftarrow \theta_{MCC} - \eta \frac{\partial \mathcal{L}}{\partial \theta_{MCC}} \tag{8}$$

where η is the learning rate, $\mathcal{L}$ is the loss function, and θ_{MCC} represents the parameters of the MCC module. The gradient calculation follows the chain rule:

$$\frac{\partial \mathcal{L}}{\partial \theta_{MCC}} = \frac{\partial \mathcal{L}}{\partial Y} \cdot \frac{\partial Y}{\partial Y_{mc}} \cdot \frac{\partial Y_{mc}}{\partial C_{mc}} \cdot \frac{\partial C_{mc}}{\partial \theta_{MCC}}. \tag{9}$$

This update forces the MCC module to learn parameters that generate a C_{mc} term, which minimizes the overall prediction error when applied to the initial prediction and bias matrix. Because meteorological variables exhibit intrinsic physical relationships in the training data, the module naturally learns these relationships through the optimization process without requiring explicit formulation of physics equations.

3.3. State Correction

The range of each variable is strictly limited based on the meteorological variables' physical characteristics; for instance, the humidity should be within $[0, 100]$. These ranges are also influenced by the location of the observation station; for example, the temperature at the North Pole should be lower than at the equator. Furthermore, certain variables are interdependent; when precipitation occurs, the humidity is typically high, and high wind velocities often occur together with lower temperatures.

These characteristics lead to strict constraints on the ranges of variable values, and result in a high risk of unreasonable states occurring during training and forecasting due to the randomness of model parameter optimizations, such as high precipitation and low humidity at the same time. Therefore, we propose that providing extra monitoring of the predicted states and correcting the unreasonable ones can lead to more reliable prediction results and more robust models.

After obtaining the constrained prediction Y_{mc}, we project Y_{mc} from the temporal domain to the hidden space by a fully connected layer f_{sp} and a nonlinear activation function σ:

$$h_s = \sigma(f_{sp}(Y_{mc})) \tag{10}$$

where $h_s \in \mathbb{R}^H$ is the representation of each state and H is the predefined dimension of the hidden space. To avoid overfitting and make the model more robust, we introduce a dropout layer after the hidden representation h_s, as previously deployed in the MCC module network:

$$h_s = Dropout(h_s). \tag{11}$$

The SC module focuses on ensuring that each predicted state adheres to the physical constraints of the meteorological variables. Unlike the MCC module, which operates based on changes in the variables, the SC module directly examines each predicted state at each time step independently. This direct projection of each time step's state into the hidden space allows the model to assess whether that state is reasonable irrespective of how it was reached.

To enhance the MLP's ability to capture the state's subtle features among the large scale of meteorological states,we introduce a differential operation on representation h_s inspired by the differential transformer [44], as demonstrated in Figure 3. This differential operation $diff$ can be calculated as follows:

$$diff(h_s) = \exp(\lambda_1) \odot h_s - \exp(\lambda_2) \odot h_s \tag{12}$$

where $\lambda_1, \lambda_2 \in \mathbb{R}^{P \times H}$ are learnable matrices and $\odot$ denotes element-wise multiplication.

Similar to a differential amplifier [45], this differential operation acts as a feature enhancement mechanism, helping to distinguish important patterns from noise in the hidden representation. By applying different learnable weights ($\exp(\lambda_1)$ and $\exp(\lambda_2)$) to the same hidden representation h_s, the operation can selectively amplify subtle but important features while suppressing common noise patterns.

Then, we deploy another fully connected layer f_{sc} to judge the state and generate a correction term C_{sc}:

$$C_{sc} = f_{sc}(h_s). \tag{13}$$

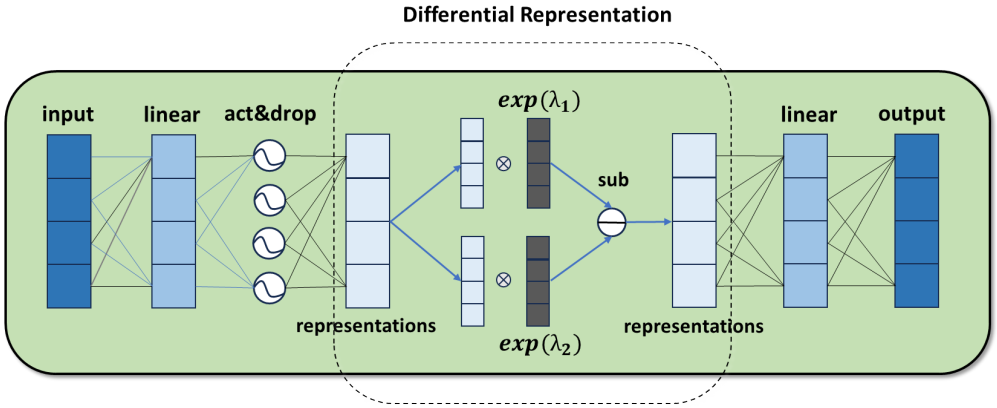


Figure 3. Architecture of the MLP with differential internal representations deployed in the state correction module. Here, λ_1, λ_2 are the learnable matrices of the differential operation on representation h_s. We introduce the differential operation $diff$ as a differential amplifier to enhance the subtle features while suppressing irrelevant noise, which helps the MLP to more effectively learn the state's subtle features among the large number of meteorological states.

Finally, we add the correction term C_{sc} to the constrained prediction Y_{mc} in order to adjust the states at each time step and generate the final prediction Y:

$$Y = Y_{mc} + C_{sc}. \tag{14}$$

Similar to the MCC module, the SC module learns through the gradient propagation mechanism during training. The gradient flow for the SC module is as follows:

$$\theta_{SC} \leftarrow \theta_{SC} - \eta \frac{\partial \mathcal{L}}{\partial \theta_{SC}} \tag{15}$$

where θ_{SC} represents the learnable parameters in the SC module, including the differential operation parameters λ_1 and λ_2. The gradient of the loss with respect to these parameters is

$$\frac{\partial \mathcal{L}}{\partial \theta_{SC}} = \frac{\partial \mathcal{L}}{\partial Y} \cdot \frac{\partial Y}{\partial C_{sc}} \cdot \frac{\partial C_{sc}}{\partial \theta_{SC}}. \tag{16}$$

Through our Equation (14), we have $\frac{\partial Y}{\partial C_{sc}} = 1$, which provides a direct gradient path from the loss to the SC module parameters. This direct gradient flow allows the SC module to efficiently learn which aspects of individual meteorological states require correction.

Therefore, the PCC module's dual-component design addresses different challenges caused by physical characteristics in weather time series forecasting. The MCC component ensures that the relationships between variables stay reasonable during the prediction process. It operates on the correlation patterns, ensuring that when one variable changes, the related variables change in correct ways. Meanwhile, the SC component ensures that the individual states at each time step remain physically plausible regardless of how they were evolved, acting as a correction mechanism that identifies and adjusts unrealistic states.

Together, these components complement each other, enabling the model to learn the variable constraints from the data without requiring explicit physical equations or laws. This makes the resulting model more adaptable and robust across varying scenarios, such as different climate regimes.

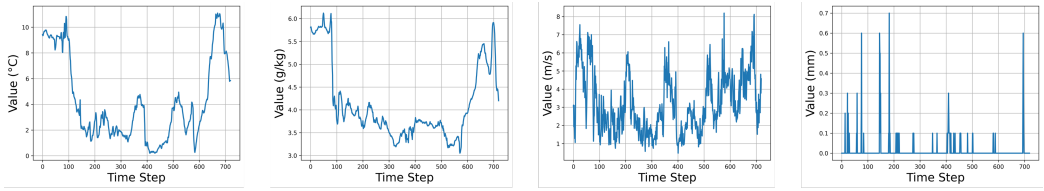
4. Experiments

4.1. Experiment Materials and Setup

4.1.1. Dataset

We conducted experiments on the weather time series dataset collected from the Max Planck Institute for Biogeochemistry, Jena, Germany in tabular format. The datasets contained 21 different meteorological variables, including temperature, humidity, and wind speed, as listed in the Appendix A. The variables were recorded every 10 min over the whole year of 2020. Using a ratio of 7:1:2, we split the dataset into a training set for training the model, a validation set for tuning the hyperparameters, and a test set for evaluating the model's performance after training.

As demonstrated in Figure 4, some of the meteorological variables exhibit unstable and nonstationary properties with noise and fluctuations, making it challenging for general forecasting models to capture useful features precisely and resulting in less effective performance on forecasting tasks.



(a) Air Temperature (b) Specific Humidity (c) Wind Velocity (d) Precipitation

Figure 4. Trends of four meteorological variables from the Max-Planck-Institut dataset for the same time period. The trends of air temperature and specific humidity show obvious similarity, while the trend of wind velocity carries dense noise and fluctuations and is relatively independent. There are also some time slots with zero precipitation.

4.1.2. Backbone Models

For our experimental evaluation, we selected three different types of time series models as the backbone models for our PCC method to serve as the baselines for the comparison. The selected backbone models have shown the state-of-the-art performance among their categories in multivariate time series forecasting tasks. The chosen models included an RNN-based model (SegRNN), two transformer-based models (iTransformer and PatchTST), and an MLP-based model (DLinear). In addition to these advanced general models, we also selected stackLSTM as a backbone model, as it has been extensively utilized in numerous weather time series research studies to model temporal variations and has demonstrated versatile performance in this regard. The detailed settings of the backbone models are listed in Appendix D.

4.1.3. Training and Evaluation Setup

During the training process, we set the default length of the observed series O to 96 (equivalent to 16 h) and set the length of the future series P as $\{96, 192, 336, 720\}$, which means predicting the weather for the next 16, 32, 56, and 120 h.

Although our PCC module is simple and lightweight compared to the backbone models, it introduces additional complexity as a supplementary component, which may introduce additional risk of unstable training. We carefully selected the hyperparameters of the PCC module as follows:

1. The hidden dimension for the two submodule networks was set as 128, balancing performance and computational complexity.
2. Tanh was chosen as the activation function σ due to its symmetry around zero and smooth gradient properties. These characteristics ensure training stability, which is

particularly important because weather variables often contain extreme values and rapid fluctuations.

3. The dropout rate was set to 0.3, representing an optimal tradeoff between regularization to prevent overfitting and maintaining sufficient network capacity.
4. The learnable matrices λ_1, λ_2 were both initialized to 1, ensuring that the differential terms start with equivalent contribution weights. This helps to prevent initial bias that may cause unstable parameter updates during the early training stages.

We evaluated PCC's sensitivity to the settings of the hidden dimension, dropout rate, and initialization of the differential matrix in Appendix C.

Our PCC module functions as a plug-and-play component that can be integrated into backbone models by simply connecting it at the feature dimension. This integration requires no modifications to the backbone architecture itself nor additional data preprocessing steps such as normalization. Therefore, in our experiments we followed the same settings for each model as in their original publications. In addition, we used the official code implementations, including data processing methods, normalization methods, hyperparameters, learning rate, etc., ensuring fairness of the comparison and reproducibility of the results. The detailed training configurations of all the backbone models are provided in Appendix D.

We evaluated the models' performance on the test set using the metrics of mean absolute error (MAE) and mean squared error (MSE). To ensure reliability, each numerical result was the average of five runs with different random seeds.

The experiments were conducted on a server with an Intel Xeon Platinum 8474C CPU, 80 GB memory, and an NVIDIA RTX 4090D GPU with 24 GB memory. The PCC module and backbone models were implemented with PyTorch 2.0.0 and CUDA 11.8 with Python 3.8 on Ubuntu 20.04.

4.2. Main Results and Discussion

4.2.1. Main Forecasting Results

As shown in Table 1, the PCC module can effectively improve the performance of all backbone models on weather time series forecasting tasks across different prediction horizons. The models augmented with the PCC module all achieve significantly lower MAE and MSE than the original models. Notably, our PCC respectively reduces the MSE of the DLinear and SegRNN models by 17.34% and 11.02% on average. The improvement is particularly pronounced for shorter prediction horizons, with a 26.57% reduction in MSE for DLinear and 11.11% reduction for SegRNN with a prediction horizon of 96. The detailed error bars, which include the standard deviation and the confidence interval, are listed in Appendix B.

Table 1. MSE and MAE of forecasting results for different backbones with and without PCC across different forecasting horizons ($\{96, 192, 336, 720\}$) with a consistent look-back length of 96. Lower value of MAE and MSE indicate better forecasting performance, and are indicated in **bold**.

Model	Design	96		192		336		720		Average	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
stackLSTM	Original	0.298	0.318	0.387	0.373	0.501	0.459	0.548	0.495	0.434	0.411
	+PCC	**0.221**	**0.266**	**0.242**	**0.289**	**0.285**	**0.314**	**0.369**	**0.376**	**0.279**	**0.311**
DLinear	Original	0.207	0.233	0.244	0.269	0.286	0.306	0.345	0.354	0.271	0.290
	+PCC	**0.152**	**0.197**	**0.196**	**0.239**	**0.243**	**0.276**	**0.304**	**0.318**	**0.224**	**0.258**

Table 1. *Cont.*

Model	Design	96		192		336		720		Average	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
iTransformer	Original	0.175	0.215	0.225	0.258	0.281	0.299	0.358	0.350	0.260	0.281
	+PCC	**0.159**	**0.204**	**0.208**	**0.248**	**0.267**	**0.291**	**0.347**	**0.344**	**0.245**	**0.272**
PatchTST	Original	0.175	0.216	0.221	0.257	0.280	0.298	0.352	0.347	0.257	0.280
	+PCC	**0.157**	**0.205**	**0.207**	**0.249**	**0.270**	**0.293**	**0.345**	**0.344**	**0.245**	**0.273**
SegRNN	Original	0.162	0.200	0.208	0.243	0.264	0.285	0.347	0.340	0.245	0.267
	+PCC	**0.144**	**0.189**	**0.189**	**0.233**	**0.238**	**0.273**	**0.299**	**0.317**	**0.218**	**0.253**

4.2.2. Different Length of Observed Time Series

For a more comprehensive evaluation, Table 2 lists the performance of the PCC module with different lengths of the observed time series. As the table shows, the PCC module improves the performance of the different backbone models across varying observation lengths. The improvements are more significant in shorter observed time series. This indicates that PCC can help to reduce the model's dependence on the length of the observed series. For instance, the DLinear model with PCC achieves better performance based on 8 h of observed time series than the original model based on 32 h of observation. Thus, our PCC module can help the model to capture the underlying patterns more effectively in the data scarcity scenario, leading to more robust and reliable results on the weather forecasting task with less data storage.

Table 2. MSE of forecasting results for different backbones with and without PCC across different lengths of observed time series ($\{48, 96, 192, 336, 720\}$) with a consistent prediction horizon of 96.

Backbone	Design	Observation Length				
		48	96	192	336	720
stackLSTM	Original	0.326	0.298	0.326	0.322	0.349
	+PCC	**0.236**	**0.221**	**0.213**	**0.195**	**0.228**
DLinear	Original	0.230	0.207	0.190	0.177	0.168
	+PCC	**0.188**	**0.152**	**0.145**	**0.143**	**0.142**
iTransformer	Original	0.202	0.175	0.170	0.162	0.175
	+PCC	**0.181**	**0.159**	**0.154**	**0.151**	**0.159**
PatchTST	Original	0.211	0.175	0.159	0.150	0.147
	+PCC	**0.188**	**0.157**	**0.150**	**0.146**	**0.145**
SegRNN	Original	0.203	0.162	0.150	0.146	0.142
	+PCC	**0.169**	**0.144**	**0.139**	**0.138**	**0.138**

Bold values indicate better results.

4.2.3. Key Variables Analysis

We also conducted a key variables analysis to demonstrate the effectiveness of the PCC module with different crucial meteorological variables, including air temperature, specific humidity, wind velocity, precipitation, etc. These variables are widely utilized in weather forecasting tasks, have shown significant impacts on climate change, and correlate closely with real-world applications such as agriculture, transportation, etc. Importantly, they also exhibit significant correlation patterns between each other and have specific ranges of values. A Table 3 demonstrates, the PCC module can improve the performance of different backbone models with respect to these different key meteorological variables.

Table 3. Prediction results of key variables using DLinear as the backbone. Results are reported as MSE and MAE, including predictions with and without PCC. The forecasting range and look-back length are both 96.

Variable	Air Temperature		Specific Humidity		Wind Velocity		Precipitation	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Original	0.099	0.223	0.097	0.212	0.001	0.021	0.067	0.055
+PCC	**0.082**	**0.218**	**0.062**	**0.173**	0.001	**0.018**	**0.052**	**0.035**

Bold values indicate better results.

4.2.4. Impact of Strongly Correlated Variables

One potential concern with the dataset used in our experiments is the presence of strongly correlated variables, particularly those related to humidity measurements (e.g., Tdew, rh, VPmax) and some temperature-related variables (e.g., T, Tlog). These variables describe similar physical properties but do so in different mathematical representations, which might introduce unwanted biases or redundancies in the model.

To investigate this concern, we conduct additional experiments using a reduced variable set that excluded these potentially redundant features. Specifically, we removed the following variables: Tdew (dew point temperature), VPmax (saturation water vapor pressure), VPact (actual water vapor pressure), VPdef (water vapor pressure deficit), sh (specific humidity), H2OC (water vapor concentration), Tpot (potential temperature), and Tlog (temperature in log).

Table 4 presents a comparison of model performance with the full variable set and the reduced variable set. Contrary to what might be expected, the models trained on the full variable set consistently outperformed those trained on the reduced set regardless of whether PCC was applied. Moreover, PCC maintained its performance improvement ability even with the reduced variable set.

Table 4. Performance comparison (MSE) of models with the full variable set versus the reduced variable set. The forecasting and look-back length are both 96.

Backbone	Design	Reduced Variable Set		Full Variable Set	
		MSE	MAE	MSE	MAE
SegRNN	Original	0.219	0.208	0.162	0.200
	+PCC	0.196	0.195	**0.144**	**0.189**
DLinear	Original	0.284	0.253	0.207	0.233
	+PCC	0.207	0.207	**0.152**	**0.197**

Bold values indicate the best results.

These results suggest that while these particular variables are mathematically related or describe similar weather conditions, they also contribute unique and valuable information that is relevant to the forecasting task. The relationships between these variables appear to provide useful signals rather than misleading correlations. Furthermore, our PCC module effectively leverages these relationships rather than being hindered by them, as evidenced by its consistent performance improvement across both variable sets.

These findings highlight the robustness and generalization capability of our approach even when processing highly correlated input features, which is a common characteristic in meteorological datasets.

4.3. Method Analysis Results and Discussion

4.3.1. Ablation Study

To verify the effectiveness of each design of the PCC module, we conducted an ablation study. We removed or replaced each part of the PCC module, including removal of the two submodules, differential representation inside the state correction network, and inverting the order of the two submodules. The experiments used SegRNN as the backbone model with a look-back window of 96 and prediction horizon of 96. As demonstrated in Table 5, the PCC module with default settings achieves the best performance. Removing the State Correction module leads to the most significant performance degradation, particularly for longer prediction horizons. This indicates that the SC module plays a crucial role in stabilizing predictions and achieving improvement, given the distinct unstable fluctuations and strict physical value constraints among the weather time series.

Table 5. Ablation results (MSE) of different PCC designs. The ablation experiments used SegRNN as the backbone model, with a look-back window of 96 and prediction horizon of 96. Here, ✓ denotes that the design is used and unchanged, w/o means that it is removed, and 'Invert' means that the order of the two modules is inverted. 'FC-only' means that the design was replaced with one fully connected layer, while 'Diff-Rep' denotes differential representation.

Design	Operation	Forecasting Length			
		96	**192**	**336**	**720**
MCC	w/o	0.149	0.193	0.241	0.302
	FC-only	0.145	0.190	0.239	0.301
SC	w/o	0.146	0.196	0.255	0.332
	FC-only	0.148	0.192	0.245	0.312
Diff-Rep	w/o	0.146	0.192	0.243	0.307
PCC	w/o	0.162	0.208	0.264	0.340
	Invert	0.147	0.192	0.244	0.309
	✓	**0.144**	**0.189**	**0.238**	**0.299**

Bold values indicate the best results.

4.3.2. Scalability Analysis

To evaluate the scalability of our approach, we conducted experiments comparing the parameter efficiency between the original backbone model and our PCC-enhanced version. We used SegRNN as the backbone model and varied its parameter scale through different hidden dimensions: {64, 128, 256, 512, 1024} for the original design and {16, 32, 64} for the PCC-enhanced design, both based on the same observation and prediction length of 96. According to the fitting curves of the results in Figure 5, the PCC-enhanced SegRNN has better accuracy with learnable parameters of less than about 1% of the original SegRNN's parameters. This indicates that our PCC helps the original forecasting models to more effectively capture crucial meteorological patterns and achieve higher accuracy with fewer parameters. This reduction in model size directly translates to lower computational demands and deployment costs on hardware devices.

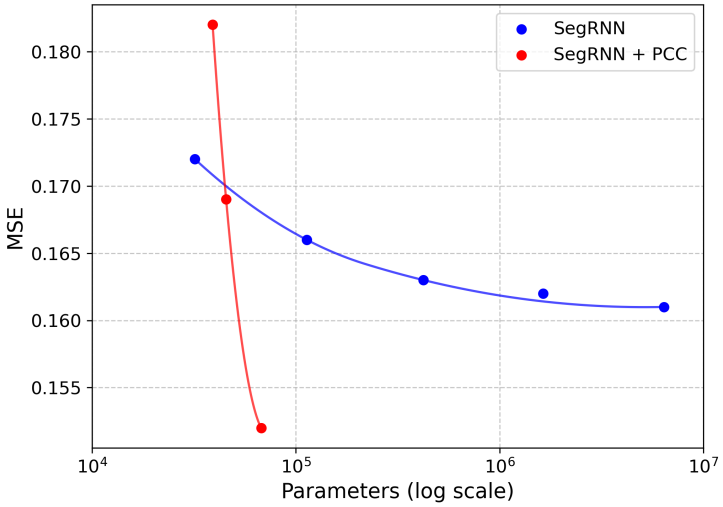


Figure 5. Fitting curves showing the performance of the SegRNN backbone model with different parameter scales with and without our PCC module. The results indicate that the proposed PCC module can help the backbone model to achieve better performance with a significantly smaller number of parameters.

4.3.3. Training Cost and Stability

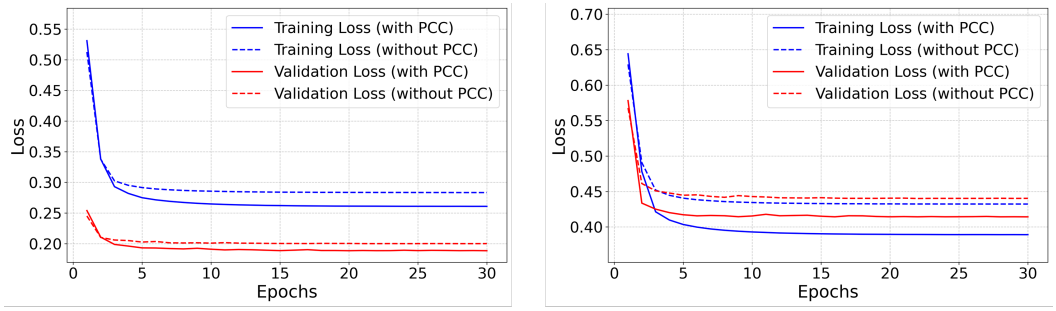
To more practically and comprehensively validate our method's efficiency and resource utilization, we present the training time and memory consumption of our method in Table 6, using SegRNN as backbone model with the same look-back and forecasting horizon of 96. The results show that the performance improvement requires only modest and acceptable increases in computational overhead. The shorter training time with the 720 horizon indicates that despite PCC introducing additional computational complexity, it can help the model to achieve more stable training and faster convergence when the forecasting horizon is longer and more likely to be unstable.

Table 6. Comparison of training time and maximum memory consumption of the SegRNN backbone with and without PCC. The look-back length is 96, while the prediction horizons are 96 and 720. We set the number of training epochs as 30 and the patience as 10.

Design	MSE		Time (s)		Memory (MB)	
	96	**720**	**96**	**720**	**96**	**720**
Original	0.162	0.347	176	227	204.55	865.39
+PCC	0.144	0.299	214	219	204.47	868.00

Notably, the memory consumption remains nearly identical (less than 0.5% difference) when the prediction horizon is 720. Counter-intuitively, the maximum memory consumption with PCC is slightly lower than the original SegRNN when the prediction horizon is 96. We attribute this to the additional influence of the PCC module on the GPU memory allocation optimization strategies of PyTorch, as the additional memory consumption introduced by PCC is too small to be a significant factor in the total memory consumption.

To evaluate the training process, Figure 6 illustrates the temporal evolution of the training and validation losses for the SegRNN backbone with and without PCC integration. The experiments were conducted with a fixed look-back window of 96 and forecasting lengths of 96 and 720. The training loss trajectories (depicted by blue lines) for both model variants exhibit stability and convergence in the latter stages of training, demonstrating that incorporation of the PCC module maintains training stability while enhancing predictive performance.



(a) Forecasting Length = 96 (b) Forecasting Length = 720

Figure 6. Evolution of the training and validation losses of SegRNN with and without PCC for forecasting lengths of 96 and 720 and with the look-back length fixed at 96.

4.3.4. Comparison with Complex Mechanisms

The computational complexity of our PCC method primarily stems from the simple fully connected layers and element-wise tensor manipulations in the differential operations. Both of these can be accelerated by parallel computing on GPUs, resulting in lower computational overhead compared to more complex structures such as self-attention mechanisms. We conducted a comparison of performance and computational cost between the proposed PCC module and some complex mechanisms, again using SegRNN as the backbone model. Table 7 shows the results. The methods for comparison included the following:

1. SAMP + SC: Replaces the MCC module with SAMP [46], a self-attention based method for capturing variable correlation in time series.
2. LIFT + SC: Replaces the MCC module with LIFT [47], a channel dependence correction method for time series forecasting using linear structures with complex calculations such as Fourier transforms.
3. saPCC: Replaces the MLP structure in both PCC submodules with a multihead self-attention mechanism.

The implementation details of these three methods are described in Appendix E.

The results show that PCC achieves superior performance with its simple linear structure while maintaining significantly lower computational complexity, making it more suitable for resource-sensitive practical scenarios. We attribute this to the temporal independence in the design of the MCC module, which avoids the negative effects of temporal variance, making it more effective in capturing physical correlation patterns.

Table 7. Performance comparison between PCC and more complex mechanisms in terms of MSE, inference time, and number of parameters. The backbone is SegRNN with a look-back length of 96 and prediction horizons of 96 and 720.

Design	MSE		Time (ms)		Parameters (k)	
	96	**720**	**96**	**720**	**96**	**720**
SAMP + SC	0.152	0.304	0.325	0.401	86.5	246.3
LIFT + SC	0.149	0.303	0.348	0.455	1694.7	2738.2
saPCC	0.147	0.301	0.311	0.411	112.5	3173.9
PCC	**0.144**	**0.299**	**0.297**	**0.375**	**35.6**	**195.4**

Bold values indicate the best performance.

4.3.5. Robustness Analysis

As deep learning models are data-driven methods, it is well known that they are sensitive to data quality. Noisy data may introduce negative impacts on model training and lead

to poor performance [48]. Unfortunately, time series data collected from weather stations are sometimes noisy [49]. In this section, we evaluate the robustness of our PCC module under various noise conditions to validate its effectiveness in practical forecasting scenarios.

First, we introduce Additive Gaussian Noise (AGN), then scale the AGN by a predefined weight and add it to the dataset. We evaluated two training cases, first adding noise only to the input sequence and then adding it to the complete sequence. We used the SegRNN model as the backbone, with the same look-back window and prediction horizon of 96. The scale weights of the AGN were 0.1 and 0.3. The results listed in Table 8 show the robustness of our PCC module against noise and demonstrate its ability to effectively capture the underlying patterns in noisy scenarios while improving the backbone model's performance.

Table 8. Results of the PCC module under different noise conditions. The backbone is SegRNN with a look-back length of 96 and prediction horizons of 96 and 720.

Noisy Input	**0.1**		**0.3**	
	MSE	**MAE**	**MSE**	**MAE**
SegRNN	0.167	0.211	0.176	0.228
SegRNN + PCC	**0.149**	**0.198**	**0.159**	**0.210**
Noisy Sequence	**0.1**		**0.3**	
	MSE	**MAE**	**MSE**	**MAE**
SegRNN	0.178	0.240	0.267	0.357
SegRNN + PCC	**0.161**	**0.228**	**0.250**	**0.346**

Bold values (except the column headers "0.1" and "0.3") indicate the better results.

4.3.6. Visualization of the Two Submodules in PCC

Figure 7 visualizes the weights and behavior of the Multi-variants Correlation Constraint (MCC) and State Correction (SC) submodules using heatmaps. We used the DLinear model as the backbone, with the same look-back and prediction length of 96.

To visualize the submodules' operational behavior while mitigating the negative impact of extreme or zero values on the visualization, we calculated the mean squared roots of the ratios between the submodules' absolute output and input values instead of using the ratios directly.

The visualizations of the weights and ratio metrics demonstrate that the two submodules have different preferences for the time steps and variables. The MCC module is more active in the early prediction stages, which are closer to the reference point, as well as for those variables with strong correlation patterns, such as humidity, wind velocity, and precipitation. The SC module places more emphasis on the later stages, following the regular pattern that the more distant points are more likely to be unstable during model prediction. In addition, it focuses more on variables which are more likely to be unstable or have more complex and significant variance patterns, such as temperature, water vapor pressure, wind velocity and direction, and photosynthetically active radiation. After the initial period, the activity of both modules becomes stable and time-insensitive.

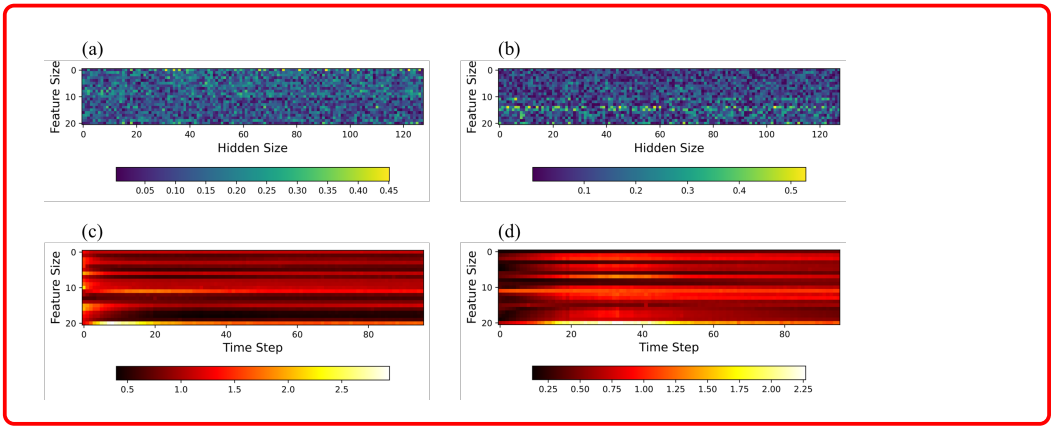


Figure 7. Heatmap visualization of the weights and input–output ratios for the MCC and SC modules within PCC. Lighter color indicates more activity. Subfigures (**a**,**b**) respectively visualize the learned weights of the MCC and SC submodules, while (**c**,**d**) show their corresponding input–output ratios. This figure demonstrates that the two submodules have different preferences for the different time steps and variables.

4.3.7. Visualization of Differential Operation

To provide more intuitive insight into the SC module's effectiveness, we directly visualize the differential operation's learnable matrices λ_1 and λ_2 in the state correction module. First, we reparameterize these two matrices as

$$\lambda_{diff} = \exp(\lambda_1) - \exp(\lambda_2), \tag{17}$$

where λ_{diff} is the reparameterized differential operation matrix. Then, we visualize λ_{diff} in Figure 8, demonstrating that the differential operation prefers to enhance those features that are farther away from the observation series.

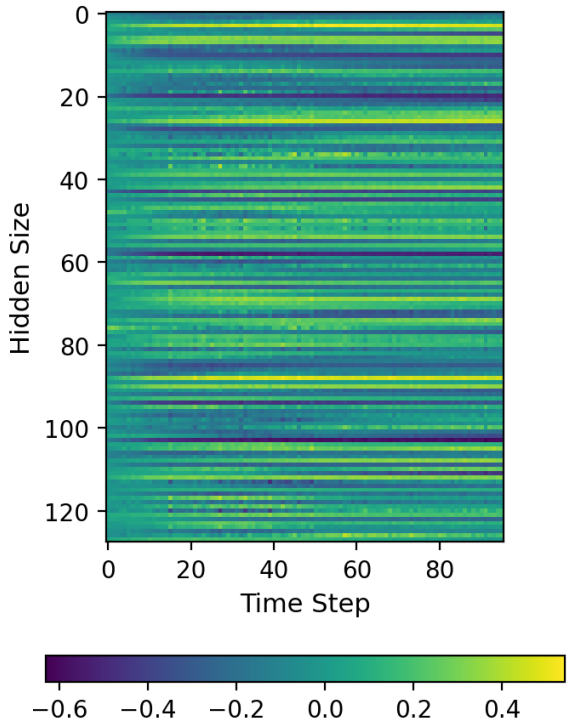


Figure 8. Visualization of the differential operation matrix acting on the hidden representation of the state correction module.

5. Conclusions and Future Work

In this paper, we have presented a generic plug-and-play approach named Post Constraint and Correction (PCC) for improving deep learning-based multivariate weather time series forecasting by explicitly incorporating domain knowledge such as the physical characteristics of meteorological variables. Our method addresses two key challenges in weather time series prediction: (1) missing or improper alignment of the variables' physical interdependencies, and (2) imprecise or unreasonable predictions caused by neural network randomness and the strict physical constraints of meteorological variables.

Extensive experimental results across different cases clearly demonstrate the effectiveness of our PCC approach in terms of improved accuracy and cost-efficiency. For instance, our method reduces the prediction MSE by up to 11% on average with the SegRNN backbone while requiring less than 0.5% additional GPU memory consumption. Our analysis of PCC's scalability demonstrates that the proposed module enables backbone models to achieve better performance with a smaller number of parameters, making it particularly suitable for resource-constrained environments where computational efficiency is as critical as forecasting accuracy.

Additionally, our ablation studies validate the importance of each component in the proposed method along with the reasonableness of our design. Visualization analysis further reveals the interpretability of the module's behavior, showing how it effectively learns and leverages the specific properties of meteorological time series to achieve improved model performance.

While our PCC module demonstrates significant performance improvements, we acknowledge interpretability limitations common to deep learning approaches in physical systems. Unlike traditional physics-based models, neural networks often function as 'black boxes', making it difficult to fully explain how they incorporate domain knowledge [50]. In our opinion, there is a promising approach to address this challenge by developing hybrid architectures that more explicitly incorporate physical relationships. For example, implementing gating mechanisms that selectively apply predefined physical constraints to the model outputs may offer an optimal balance between purely data-driven approaches and physics-constrained models. Such hybrid approaches can not only enhance interpretability but might further reduce model complexity while maintaining or improving prediction accuracy. We consider this a promising direction for our ongoing research, and plan to thoroughly investigate such mechanisms in subsequent work.

Author Contributions: Conceptualization, Z.W., Z.L. and Y.L.; methodology, Z.W.; software, Z.W.; validation, Z.W.; formal analysis, Z.W.; investigation, Z.W. and Z.Y.; resources, Z.W. and Z.L.; data curation, Z.W. and Y.L.; writing—original draft preparation, Z.W.; writing—review and editing, Z.W., Z.L. and Z.Y.; visualization, Z.W. and Z.Y.; supervision, Z.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The dataset we used in our experiment was obtained from the open time series library at https://github.com/thuml/Time-Series-Library (accessed on 29 May 2024). This dataset is preprocessed in a tabular format. The raw CSV files for different years are available at https://www.bgc-jena.mpg.de/wetter/weather_data.html (accessed on 29 May 2024).

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Dataset Variables

In this appendix, we provide a detailed list of all variables used in the Max-Planck-Institut Weather Time Series dataset. The dataset contains 21 meteorological parameters. The following table presents the symbol, unit, and detailed description of each variable.

Table A1. Variables and descriptions of the Max-Planck-Institut Weather Time Series dataset.

Symbol	Unit	Description
P	mbar	air pressure
T	°C	air temperature
Tpot	K	potential temperature
Tdew	°C	dew point temperature
rh	%	relative humidity
VPmax	mbar	saturation water vapor pressure
VPact	mbar	actual water vapor pressure
VPdef	mbar	water vapor pressure deficit
sh	g/kg	specific humidity
H2OC	mmol/mol	water vapor concentration
rho	g/m^3	air density
wv	m/s	wind velocity
max. wv	m/s	maximum wind velocity
wd	°	wind direction
rain	mm	precipitation
raining	s	precipitation duration
SWDR	W/m^2	shortwave downward radiation
PAR	W/m^2	photosynthetically active radiation
max. PAR	W/m^2	maximum photosynthetically active radiation
Tlog	°C	temperature in log
CO_2	ppm	carbon dioxide concentration of ambient air

Appendix B. Error Bars

To validate our method's robustness against initialization and random factors, Table A2 presents the standard deviations and confidence intervals of the five backbone models with and without PCC. All results are calculated across five runs with different random seeds.

Table A2. Standard deviations and confidence intervals of the five backbone models with and without PCC.

stack LSTM	+PCC		Original		Confidence Interval
	MSE	MAE	MSE	MAE	
96	0.221 ± 0.016	0.266 ± 0.011	0.298 ± 0.026	0.318 ± 0.017	99%
192	0.242 ± 0.009	0.289 ± 0.008	0.387 ± 0.014	0.373 ± 0.012	99%
336	0.285 ± 0.009	0.314 ± 0.006	0.501 ± 0.048	0.459 ± 0.038	99%
720	0.369 ± 0.016	0.376 ± 0.012	0.548 ± 0.004	0.495 ± 0.004	99%
DLinear	**+PCC**		**Original**		**Confidence Interval**
	MSE	**MAE**	**MSE**	**MAE**	
96	0.152 ± 0.000	0.197 ± 0.001	0.207 ± 0.000	0.233 ± 0.000	99%
192	0.196 ± 0.000	0.239 ± 0.000	0.244 ± 0.000	0.269 ± 0.000	99%
336	0.243 ± 0.000	0.276 ± 0.001	0.286 ± 0.000	0.306 ± 0.000	99%
720	0.304 ± 0.001	0.318 ± 0.001	0.345 ± 0.000	0.354 ± 0.001	99%

Table A2. *Cont.*

iTransformer	+PCC		Original		Confidence Interval
	MSE	MAE	MSE	MAE	
96	0.159 ± 0.001	0.204 ± 0.001	0.175 ± 0.001	0.215 ± 0.001	99%
192	0.208 ± 0.001	0.248 ± 0.001	0.225 ± 0.001	0.258 ± 0.001	99%
336	0.267 ± 0.001	0.291 ± 0.001	0.281 ± 0.001	0.299 ± 0.001	99%
720	0.347 ± 0.001	0.344 ± 0.001	0.358 ± 0.001	0.350 ± 0.001	99%
PatchTST	**+PCC**		**Original**		**Confidence Interval**
	MSE	**MAE**	**MSE**	**MAE**	
96	0.157 ± 0.001	0.205 ± 0.001	0.175 ± 0.001	0.216 ± 0.001	99%
192	0.207 ± 0.001	0.249 ± 0.001	0.221 ± 0.001	0.257 ± 0.002	99%
336	0.270 ± 0.001	0.293 ± 0.001	0.280 ± 0.001	0.298 ± 0.001	99%
720	0.345 ± 0.001	0.344 ± 0.001	0.352 ± 0.001	0.347 ± 0.001	99%
SegRNN	**+PCC**		**Original**		**Confidence Interval**
	MSE	**MAE**	**MSE**	**MAE**	
96	0.144 ± 0.000	0.189 ± 0.000	0.162 ± 0.000	0.200 ± 0.000	99%
192	0.189 ± 0.001	0.233 ± 0.001	0.208 ± 0.000	0.243 ± 0.000	99%
336	0.238 ± 0.001	0.273 ± 0.001	0.264 ± 0.000	0.285 ± 0.000	99%
720	0.299 ± 0.001	0.317 ± 0.001	0.347 ± 0.000	0.340 ± 0.000	99%

Appendix C. Hyperparameter Sensitivity

We evaluated the sensitivity of our method to key hyperparameters, including the hidden dimensions and dropout rate of the two submodules' networks. The results are shown in Figure A1, and the initialization of the SC module's differential matrices are shown in Table 2. The backbone model is SegRNN with a look-back length of 96. These results demonstrate that our method's performance is insensitive and robust across different hyperparameter settings, suggesting efficient hyperparameter tuning costs during training.

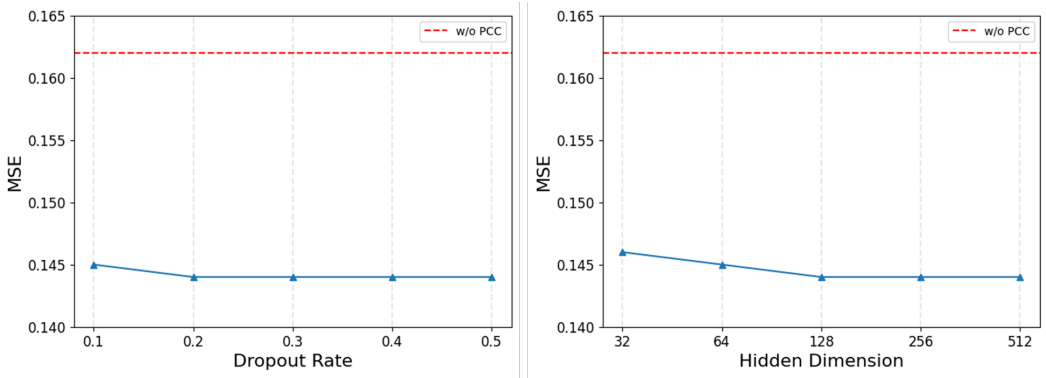


Figure A1. Performance of our method with different hyperparameter settings, including the hidden dimensions and dropout rates of the submodule networks. SegRNN is the backbone model, with a look-back length of 96 and prediction horizon of 96.

Table A3. MSE of forecasting results for SegRNN + PCC with different initialization of the SC submodule's differential matrices across different lengths of forecasting series ({96, 192, 336, 720}) with a consistent observation length of 96.

Setting	MSE			
	96	192	336	720
One	0.144 ± 0.000	0.189 ± 0.001	0.238 ± 0.001	0.299 ± 0.001
Zero	0.144 ± 0.000	0.190 ± 0.000	0.239 ± 0.001	0.299 ± 0.001
Zero-One	0.145 ± 0.001	0.190 ± 0.001	0.240 ± 0.001	0.300 ± 0.001
One-Zero	0.145 ± 0.001	0.190 ± 0.001	0.240 ± 0.001	0.302 ± 0.002

Appendix D. Detailed Settings of Backbone Models

We chose five different models as backbone models for comparison: SegRNN, iTransformer, PatchTST, DLinear, and stackLSTM. The detailed settings used for the backbone models are listed in Tables A4–A8.

Table A4. Detailed hyperparameter settings of SegRNN.

Parameter	Value	Parameter	Value
Segment Length	48	Learning rate	0.0001
RNN type	GRU	Batch size	64
RNN layers	1	Training epochs	30
Hidden size	512	Training patience	10
Dropout	0.5	Training loss	MAE

Table A5. Detailed hyperparameter settings of iTransformer.

Parameter	Value	Parameter	Value
Encoder layers	3	Learning rate	0.0001
Heads	8	Batch size	32
Hidden dimensions	512	Training epochs	10
Dropout rate	0.1	Training patience	3
Training loss	MSE		

Table A6. Detailed hyperparameter settings of PatchTST.

Parameter	Value	Parameter	Value
Patch size	16	Learning rate	0.0001
Stride	8	Batch size (96, 192)	32
Head (96, 336, 720)	4	Batch size (336, 720)	128
Head (192)	16	Training epochs	3
Hidden dimensions	512	Training patience	3
Encoder layers	2	Training loss	MSE

Table A7. Detailed hyperparameter settings of DLinear.

Parameter	Value	Parameter	Value
Kernel size	25	Learning rate	0.0001
Training epochs	30	Batch size	64
Training patience	10	Training loss	MAE

Table A8. Detailed hyperparameter settings of stackLSTM.

Parameter	Value	Parameter	Value
Hidden size	512	Learning rate	0.0001
LSTM layers	4	Batch size	64
Training epochs	30	Training loss	MAE
Training patience	10		

Appendix E. Implementation Details of Complex Mechanisms

In this manuscript's Section 4.1, we chose three different complex mechanisms for comparison with our PCC module: SAMP, saPCC, and LIFT. Of these, SAMP and saPCC are both based on multi-head self-attention, while LIFT is based on a linear structure. For the attention-based methods, we set the number of attention heads in SAMP to 1 and that in saPCC to 8. For LIFT, we set the number of leaders to 4, number of states to 8, and temperature to 1.0.

References

1. Graham, A.; Mishra, E.P. Time series analysis model to forecast rainfall for Allahabad region. *J. Pharmacogn. Phytochem.* **2017**, *6*, 1418–1421.
2. Shivhare, N.; Rahul, A.K.; Dwivedi, S.B.; Dikshit, P.K.S. ARIMA based daily weather forecasting tool: A case study for Varanasi. *Mausam* **2019**, *70*, 133–140.
3. Poterjoy, J. Implications of multivariate non-Gaussian data assimilation for multiscale weather prediction. *Mon. Weather. Rev.* **2022**, *150*, 1475–1493.
4. Moreno, S.R.; dos Santos Coelho, L. Wind speed forecasting approach based on singular spectrum analysis and adaptive neuro fuzzy inference system. *Renew. Energy* **2018**, *126*, 736–754. [CrossRef]
5. Yano, J.I.; Ziemiański, M.Z.; Cullen, M.; Termonia, P.; Onvlee, J.; Bengtsson, L.; Carrassi, A.; Davy, R.; Deluca, A.; Gray, S.L.; et al. Scientific challenges of convective-scale numerical weather prediction. *Bull. Am. Meteorol. Soc.* **2018**, *99*, 699–710.
6. Schultz, M.G.; Betancourt, C.; Gong, B.; Kleinert, F.; Langguth, M.; Leufen, L.H.; Mozaffari, A.; Stadtler, S. Can deep learning beat numerical weather prediction? *Philos. Trans. R. Soc.* **2021**, *379*, 20200097.
7. Wang, Y.; Wu, H.; Dong, J.; Liu, Y.; Long, M.; Wang, J. Deep time series models: A comprehensive survey and benchmark. *arXiv* **2024**, arXiv:2407.13278.
8. Lim, B.; Zohren, S. Time-series forecasting with deep learning: A survey. *Philos. Trans. R. Soc. A* **2021**, *379*, 20200209.
9. Wilby, R.L.; Troni, J.; Biot, Y.; Tedd, L.; Hewitson, B.C.; Smith, D.M.; Sutton, R.T. A review of climate risk information for adaptation and development planning. *Int. J. Climatol. J. R. Meteorol. Soc.* **2009**, *29*, 1193–1215.
10. Bauer, P.; Thorpe, A.; Brunet, G. The quiet revolution of numerical weather prediction. *Nature* **2015**, *525*, 47–55.
11. Shen, C. A transdisciplinary review of deep learning research and its relevance for water resources scientists. *Water Resour. Res.* **2018**, *54*, 8558–8593.
12. Kurth, T.; Subramanian, S.; Harrington, P.; Pathak, J.; Mardani, M.; Hall, D.; Miele, A.; Kashinath, K.; Anandkumar, A. Fourcastnet: Accelerating global high-resolution weather forecasting using adaptive fourier neural operators. In Proceedings of the Platform for Advanced Scientific Computing Conference, Davos, Switzerland, 26–28 June 2023; pp. 1–11.
13. Zhu, X.; Xiong, Y.; Wu, M.; Nie, G.; Zhang, B.; Yang, Z. Weather2k: A multivariate spatio-temporal benchmark dataset for meteorological forecasting based on real-time observation data from ground weather stations. *arXiv* **2023**, arXiv:2302.10493.
14. Dubey, A.K.; Kumar, A.; García-Díaz, V.; Sharma, A.K.; Kanhaiya, K. Study and analysis of SARIMA and LSTM in forecasting time series data. *Sustain. Energy Technol. Assessments* **2021**, *47*, 101474. [CrossRef]
15. Ray, S.; Das, S.S.; Mishra, P.; Al Khatib, A.M.G. Time series SARIMA modelling and forecasting of monthly rainfall and temperature in the South Asian countries. *Earth Syst. Environ.* **2021**, *5*, 531–546. [CrossRef]
16. Hewage, P.; Behera, A.; Trovati, M.; Pereira, E.; Ghahremani, M.; Palmieri, F.; Liu, Y. Temporal convolutional neural (TCN) network for an effective weather forecasting using time-series data from the local weather station. *Soft Comput.* **2020**, *24*, 16453–16482. [CrossRef]
17. Verdonck, T.; Baesens, B.; Óskarsdóttir, M.; vanden Broucke, S. Special issue on feature engineering editorial. *Mach. Learn.* **2024**, *113*, 3917–3928. [CrossRef]
18. Bi, K.; Xie, L.; Zhang, H.; Chen, X.; Gu, X.; Tian, Q. Accurate medium-range global weather forecasting with 3D neural networks. *Nature* **2023**, *619*, 533–538. [CrossRef]

19. Chen, K.; Han, T.; Gong, J.; Bai, L.; Ling, F.; Luo, J.J.; Chen, X.; Ma, L.; Zhang, T.; Su, R.; et al. Fengwu: Pushing the skillful global medium-range weather forecast beyond 10 days lead. *arXiv* **2023**, arXiv:2304.02948.
20. Karevan, Z.; Suykens, J.A. Transductive LSTM for time-series prediction: An application to weather forecasting. *Neural Netw.* **2020**, *125*, 1–9. [CrossRef]
21. Al Sadeque, Z.; Bui, F.M. A deep learning approach to predict weather data using cascaded LSTM network. In Proceedings of the 2020 IEEE Canadian conference on electrical and computer engineering (CCECE), London, ON, Canada, 30 August–2 September 2020; IEEE: New York, NY, USA, 2020; pp. 1–5.
22. Dikshit, A.; Pradhan, B.; Alamri, A.M. Long lead time drought forecasting using lagged climate variables and a stacked long short-term memory model. *Sci. Total Environ.* **2021**, *755*, 142638.
23. Yan, Z.; Lu, X.; Wu, L. Exploring the Effect of Meteorological Factors on Predicting Hourly Water Levels Based on CEEMDAN and LSTM. *Water* **2023**, *15*, 3190. [CrossRef]
24. Wang, H. Weather temperature prediction based on LSTM and transformer. In Proceedings of the International Conference on Electronics, Electrical and Information Engineering (ICEEIE 2024), Bangkok ,Thailand, 16–18 August 2024; SPIE: Bellingham, WA, USA, 2024; Volume 13445, pp. 206–214.
25. Sezer, O.B.; Gudelek, M.U.; Ozbayoglu, A.M. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Appl. Soft Comput.* **2020**, *90*, 106181.
26. Zheng, J.; Huang, M. Traffic flow forecast through time series analysis based on deep learning. *IEEE Access* **2020**, *8*, 82562–82570.
27. Jaseena, K.; Kovoor, B.C. Deterministic weather forecasting models based on intelligent predictors: A survey. *J. King Saud Univ.-Comput. Inf. Sci.* **2022**, *34*, 3393–3412.
28. Xiao, J.; Zhou, Z. Research progress of RNN language model. In Proceedings of the 2020 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA), Dalian, China, 27–29 June 2020; IEEE: New York, NY, USA, 2020; pp. 1285–1288.
29. Hewamalage, H.; Bergmeir, C.; Bandara, K. Recurrent neural networks for time series forecasting: Current status and future directions. *Int. J. Forecast.* **2021**, *37*, 388–427.
30. Saini, U.; Kumar, R.; Jain, V.; Krishnajith, M. Univariant Time Series forecasting of Agriculture load by using LSTM and GRU RNNs. In Proceedings of the 2020 IEEE Students Conference on Engineering & Systems (SCES), Prayagraj, India, 10–12 July 2020; IEEE: New York, NY, USA, 2020; pp. 1–6.
31. Casado-Vara, R.; Martin del Rey, A.; Pérez-Palau, D.; de-la Fuente-Valentín, L.; Corchado, J.M. Web traffic time series forecasting using LSTM neural networks with distributed asynchronous training. *Mathematics* **2021**, *9*, 421. [CrossRef]
32. Amalou, I.; Mouhni, N.; Abdali, A. Multivariate time series prediction by RNN architectures for energy consumption forecasting. *Energy Rep.* **2022**, *8*, 1084–1091.
33. Lin, S.; Lin, W.; Wu, W.; Zhao, F.; Mo, R.; Zhang, H. Segrnn: Segment recurrent neural network for long-term time series forecasting. *arXiv* **2023**, arXiv:2308.11200.
34. Wen, Q.; Zhou, T.; Zhang, C.; Chen, W.; Ma, Z.; Yan, J.; Sun, L. Transformers in time series: A survey. *arXiv* **2022**, arXiv:2202.07125.
35. Ren, H.; Dai, H.; Dai, Z.; Yang, M.; Leskovec, J.; Schuurmans, D.; Dai, B. Combiner: Full attention transformer with sparse computation cost. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 22470–22482.
36. Nie, Y.; Nguyen, N.H.; Sinthong, P.; Kalagnanam, J. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv* **2022**, arXiv:2211.14730.
37. Liu, Y.; Hu, T.; Zhang, H.; Wu, H.; Wang, S.; Ma, L.; Long, M. itransformer: Inverted transformers are effective for time series forecasting. *arXiv* **2023**, arXiv:2310.06625.
38. Wu, H.; Xu, J.; Wang, J.; Long, M. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 22419–22430.
39. Liu, Y.; Wu, H.; Wang, J.; Long, M. Non-stationary transformers: Exploring the stationarity in time series forecasting. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 9881–9893.
40. Zeng, A.; Chen, M.; Zhang, L.; Xu, Q. Are transformers effective for time series forecasting? In Proceedings of the AAAI Conference on Artificial Intelligence, Washington, DC, USA, 7–14 February 2023; Volume 37, pp. 11121–11128.
41. Yun, K.S.; Lee, J.Y.; Timmermann, A.; Stein, K.; Stuecker, M.F.; Fyfe, J.C.; Chung, E.S. Increasing ENSO–rainfall variability due to changes in future tropical temperature–rainfall relationship. *Commun. Earth Environ.* **2021**, *2*, 43.
42. Wilby, R.L.; Wigley, T. Precipitation predictors for downscaling: Observed and general circulation model relationships. *Int. J. Climatol. J. R. Meteorol. Soc.* **2000**, *20*, 641–661.
43. Park, S.; Kwak, N. Analysis on the dropout effect in convolutional neural networks. In Proceedings of the Computer Vision–ACCV 2016: 13th Asian Conference on Computer Vision, Taipei, Taiwan, 20–24 November 2016; Revised Selected Papers, Part II 13; Springer: Cham, Switzerland, 2017; pp. 189–204.
44. Ye, T.; Dong, L.; Xia, Y.; Sun, Y.; Zhu, Y.; Huang, G.; Wei, F. Differential transformer. *arXiv* **2024**, arXiv:2410.05258.

45. Laplante, P.A.; Cravey, R.; Dunleavy, L.P.; Antonakos, J.L.; LeRoy, R.; East, J.; Buris, N.E.; Conant, C.J.; Fryda, L.; Boyd, R.W.; et al. *Compr. Dict. Electr. Eng.*; CRC Press: Boca Raton, FL, USA, 2018.
46. Wang, H.; Wang, Z.; Niu, Y.; Liu, Z.; Li, H.; Liao, Y.; Huang, Y.; Liu, X. An Accurate and interpretable framework for trustworthy process monitoring. *IEEE Trans. Artif. Intell.* **2023**, *5*, 2241–2252.
47. Zhao, L.; Shen, Y. Rethinking Channel Dependence for Multivariate Time Series Forecasting: Learning from Leading Indicators. *arXiv* **2024**, arXiv:2401.17548.
48. Whang, S.E.; Lee, J.G. Data collection and quality challenges for deep learning. *Proc. VLDB Endow.* **2020**, *13*, 3429–3432.
49. Zhong, R.; Jun, S.; Xu, P. Analysis and de-noise of time series data from automatic weather station using chaos-based adaptive B-spine method. In Proceedings of the 2011 International Conference on Remote Sensing, Environment and Transportation Engineering, Nanjing, China, 24–26 June 2011; IEEE: New York, NY, USA, 2011; pp. 4765–4769.
50. Yang, R.; Hu, J.; Li, Z.; Mu, J.; Yu, T.; Xia, J.; Li, X.; Dasgupta, A.; Xiong, H. Interpretable machine learning for weather and climate prediction: A review. *Atmos. Environ.* **2024**, *338*, 120797.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.