Journal of Computational Science 99 (2026) 102936

Contents lists available at ScienceDirect

Journal of Computational Science

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

Interpretable machine learning for shared feature identification and time series prediction

Yuanlin Gu [a,*], Mao Zhang [b], Baihua Li [c], Qinggang Meng [c]

[a] Division of Computer Science & Maths, University of Stirling, Stirling FK9 4LA, United Kingdom
[b] Business School, University of St Andrews, St-Andrews KY16 9AJ, United Kingdom
[c] Computer Science Department, Loughborough University, Loughborough LE11 3TU, United Kingdom

ARTICLE INFO

Keywords:
Interpretable machine learning
Time series prediction
Model structure detection

ABSTRACT

When building a predictive model with sub-datasets from diverse locations, scenarios, or participants, a single model may not capture the unique characteristics of each sub-dataset. However, creating individual models for each dataset can be time-consuming and may overlook shared features.

In this article, a Common Structure Neural Network (CSNN) model is introduced to address these issues. The model includes a new feature selection layer that identifies critical shared factors influencing multiple outputs, allowing for a shared model structure and reduced training costs, while accurately representing the diversity within each sub-dataset.

The effectiveness of the model is demonstrated through one simulation and two real-world case studies on air pollution and stock prices. The experiments show that the model improves prediction accuracy and efficiency compared to other methods. Additionally, it enhances interpretability by revealing correlations and interactions across different locations, offering valuable insights.

1. Introduction

Constructing a model that accurately represents a complex nonlinear system involves two primary approaches. The first method involves establishing mathematical representations based on the underlying physical rules. However, manual identification of the correct model structures through experimentation can be challenging [1]. Consequently, a black box modelling approach is often employed to automatically discern the suitable model structure and parameters capable of describing the unknown components of the system.

In recent years, various data-driven modelling techniques have been developed and applied across different fields, including neural networks [2-6], fuzzy models [7], and polynomial models [8-11]. Neural networks and their variants have shown impressive accuracy in prediction and classification tasks [12]. However, the complexity of these models, with numerous hidden layers and neurons, makes it essential to interpret the model structure and understand the uncertainty involved [13]. This is especially important for ensuring the reliability of models used in critical situations, such as predicting peak values [14]. As a result, there is an urgent need to develop interpretable predictive models, a challenge that becomes even more significant given that real-world data is often imbalanced.

One key method for interpreting deep neural networks involves post-hoc explanations, such as simplifying the model, analysing feature relevance or importance, and using visual explanations [15]. In complex deep neural networks, feature importance analysis has become an essential way to enhance interpretability and provide human-understandable insights by ranking and evaluating how much each feature influences predictions [16,17]. Shapley Additive Explanations (SHAP) is a popular method for this, as it clarifies machine learning models by measuring the contribution of individual features to predictions [18]. Another approach is to create hybrid models that combine an interpretable model with a deep neural network [13].

Traditional regression-based methods, like polynomials, use simpler and more intuitive structures. While these models are easy to interpret, they may not perform as well in prediction tasks, especially those involving peak values and high-dimensional, noisy data. However, advances have led to the development of more sophisticated regression models for various prediction scenarios. The Nonlinear AutoRegressive Moving Average with eXogenous input (NARMAX) model and its variants are designed for data modelling in temporal, frequency, and spatiotemporal domains [11]. It can autonomously generate nonlinear

* Corresponding author.
E-mail addresses: yuanlin.gu@stir.ac.uk (Y. Gu), mz71@st-andrews.ac.uk (M. Zhang), b.li@lboro.ac.uk (B. Li), q.meng@lboro.ac.uk (Q. Meng).

https://doi.org/10.1016/j.jocs.2026.102936
Received 4 April 2025; Received in revised form 6 May 2026; Accepted 8 June 2026
Available online 11 June 2026

features with meaningful physical interpretations and identify the most impactful ones to build a transparent model. Typically, the initial feature space is large, and including all features can lead to time-consuming training and potential overfitting. The term selection process within NARMAX is effective for real-world applications involving large datasets. This model and its variations have been successfully applied in fields like space weather [19], environmental modelling [20], healthcare [21], and more.

These advancements have solved many challenges in data-driven modelling, but a key challenge remains in applying these methods to tasks involving numerous sub-datasets from various locations, scenarios, or participants. In such cases, a single model may not capture the unique characteristics of each sub-dataset, while creating separate models for each one can overlook shared features and miss important insights, such as spatial effects. This is particularly acceptable since studies have shown that considering spatial effects is crucial in certain time series prediction applications, like when air pollution at one location affects nearby areas [22]. Additionally, efficiently using all available data while minimizing time and computational demands has become increasingly difficult. To tackle this, a shared model structure with different parameters for each sub-dataset can be used to estimate multiple outputs simultaneously. This approach reduces model complexity and training costs while effectively representing the system and uncovering shared dynamics across diverse datasets [23].

In general, various data-driven models have been developed to address this issue. For instance, a deep learning method was developed for traffic speed forecasting, incorporating a common layer to predict multiple outputs simultaneously [24]. A regression approach was used to forecast concentrations while also identifying shared information among similar individuals [25]. Additionally, a temporal convolutional neural network was designed to improve passenger demand forecasting, outperforming other contemporary models [26]. Fu et al. (2024) proposed the CTF-former, a simplified framework combining convolution and attention mechanisms for joint temporal–cross-variable modelling [27]. Deng et al. (2023) introduced a multi-view approach that leverages spatial and temporal perspectives to improve forecasting efficiency [28]. Wu and Peng (2022) developed Pre-SMATS, which enhances generalization in small multi-stage seasonal datasets through joint feature extraction and stage classification [29]. Wei et al. (2023) applied an LSTM-based multi-task framework for ultra-short-term wind power forecasting, showing that auxiliary tasks can improve predictive accuracy and robustness [30]. These advancements highlight the effectiveness of predictive models in handling prediction tasks involving multiple outputs.

However, there are existing challenges. Most of the existing methods use standard components with shared layers but fail to recognize shared features across different subjects. Standard components with shared layers refer to neural network architectures where the same model structure is applied across different tasks or datasets to capture generalizable patterns. However, while these methods may use a shared backbone, they often do not explicitly select or emphasize shared features across different subjects which are input features that consistently influence outputs across multiple sub-datasets (e.g., different locations, individuals, or companies). These shared features may reflect underlying common dynamics or structural similarities among the subjects, such as environmental or economic factors that impact all groups similarly. Existing methods tend to optimize for overall prediction accuracy without specifically identifying which features are commonly relevant across tasks. As a result, the models may replace task-specific patterns with general ones, leading to reduced interpretability and suboptimal generalization. Furthermore, without dedicated mechanisms for shared feature identification, these models are less effective in extracting meaningful cross-subject insights or reducing model complexity through informed feature selection. This lack of insight into these features makes it difficult to explain the common factors affecting all outputs. While polynomial models can improve interpretability, they often perform worse than deep networks [20]. Therefore, it is essential to develop a deep learning approach that can accurately identify both common model structures and features while maintaining high performance.

Based on the above, this paper presents a Common Structure Neural Network (CSNN) model. The effectiveness is demonstrated through various case studies. First, we show a simple example using simulated data. Then, we apply the model to real-world problems, specifically air pollution prediction and stock price analysis. The results highlight how the CSNN model performs and compares with other machine learning methods. The main contributions of the proposed method are:

- The model identifies shared features that significantly affect outcomes across different datasets and constructs a unified model structure that simultaneously predicts outputs from multiple datasets, greatly reducing model complexity and training costs.
- Using these shared factors, the model applies a cost-effective separate parameter estimation process for each sub-dataset, enhancing prediction performance for each individual dataset.

The paper is organized as follows: Section 2 reviews related work in the field, providing context and background for the study. Section 3 introduces the proposed methods, detailing the approach and techniques used. Section 4 presents the experimental results from three case studies, demonstrating the application and effectiveness of the methods. Finally, Section 5 provides conclusions based on the findings, summarizing the key insights and implications of the research.

2. Feature identification for single dataset

For a general prediction task, consider the input and output variables described as:

$$\boldsymbol{x} = \begin{bmatrix} x_r(1) & x_r(1) & \cdots & x_r(1) \\ x_r(2) & x_r(2) & & x_r(2) \\ \vdots & & \ddots & \vdots \\ x_r(N) & x_r(N) & \cdots & x_r(N) \end{bmatrix}. \tag{1}$$

$$\boldsymbol{y} = \begin{bmatrix} y\ (1) \\ y\ (2) \\ \vdots \\ y\ (N) \end{bmatrix} \tag{2}$$

where 'r' is the index of the inputs ($r = 1, 2, \ldots R$) and N represent the numbers of data samples. The objective of the modelling task is to predict the output $y\ (t)$ from the inputs $x_r(t)$.

In many instances, this predictive model relies on both the linear variables and the extra generated features. This is because linear input variables alone are insufficient in representing the nonlinear dynamics. The derived nonlinear features offer a more accurate description of the system dynamics, particularly in the case of nonlinear complex dynamic systems. Hence, incorporating nonlinear features becomes essential to construct models that achieve higher prediction performance.

Various approaches have been developed to derive extra features that can represent complex nonlinear systems. For example, convolutional layer is commonly employed as the first component of a deep learning model [31]; the autoencoder usually works with recurrent neural network for time series prediction [32]; nonlinear terms are often generated at the first stage of establishing a NARMAX model [9].

For time series analysis in particular, the long-term time dependencies need to be captured by the established model. For example, recurrent neural networks use gates to extract relevant information from sequences [32]. Based on our prior research, we also found that pre-defined time lags can significantly reduce training time while

preserving prediction accuracy [13].

Following our previous work in [13], the linear features with time lags can be defined as follows:

$$\begin{aligned}\boldsymbol{\varphi}_{linear} = \{&y(t-D),\ldots, \quad y(t-D-d_y), \quad x_1(t-D)\ldots, \quad x_1(t-D-d_x),\\ &x_2(t-D),\ldots, \quad x_2(t-D-d_x) \quad \ldots, \quad x_R(t-D), \quad \ldots,\\ &x_R(t-D-d_x)\}\end{aligned} \tag{3}$$

where D is the time delay; d_x, and d_y are the time lags for inputs and outputs respectively.

To effectively model the complex nonlinear system, extra derived nonlinear features are found to be beneficial, and as a result, certain methods are applied to generate them. For example, some basic features with nonlinear degree of 2 and 3 can be defined as:

$$\{\forall A_i \times \forall A_j, \quad \forall A_i \times \forall A_j \times \forall A_o\} \tag{4}$$

where A_i, A_j and A_o are elements of the set $\{\boldsymbol{\varphi}^{linear}\}$. Some additional nonlinear conversion methods include reciprocal transformations denoted by $1/\forall A_i$, square root transformations denoted by $\sqrt{\forall A_i}$ and others. We denote these generated features through the above procedure as $\boldsymbol{\varphi}_{nonlinear}$. The complete feature set is then defined as:

$$\boldsymbol{\varphi} = \{\boldsymbol{\varphi}^{linear}, \boldsymbol{\varphi}^{nonlinear}, C\} \tag{5}$$

where C represents a constant.

Based on the above steps, features of all the sub-datasets can be produced, denoted as $\boldsymbol{\varphi}^{(k)}$ for $k = 1, 2, \ldots, K$. Each feature set contains M features, and the features specific to the k-th sub-dataset are $\{\boldsymbol{\varphi}_1^{(k)}, \boldsymbol{\varphi}_2^{(k)} \ldots, \boldsymbol{\varphi}_M^{(k)}\}$. For example, consider a single-input system with $D = 1$, $d_x = 1$, $d_y = 1$ and nonlinear features are generated by deriving interaction term. The features specific to the k-th sub-dataset are

$$\begin{aligned}&y^{(k)}(t-1), \quad y^{(k)}(t-2), \quad x^{(k)}(t-1), \quad x^{(k)}(t-2), y^{(k)}(t-1)\\ &\times y^{(k)}(t-2), \quad y^{(k)}(t-1) \times x^{(k)}(t-1), \quad y^{(k)}(t-1)\\ &\times x^{(k)}(t-2), \quad y^{(k)}(t-2) \times x^{(k)}(t-1), y^{(k)}(t-2)\\ &\times x^{(k)}(t-2), \quad x^{(k)}(t-1) \times x^{(k)}(t-2)\end{aligned} \tag{6}$$

The model can be further expressed as follows:

$$\begin{aligned}y^{(k)}(t) = f\,[&y^{(k)}(t-1), \quad y^{(k)}(t-2), \quad x^{(k)}(t-1), \quad x^{(k)}(t-2), y^{(k)}(t-1)\\ &\times y^{(k)}(t-2), \quad y^{(k)}(t-1) \times x^{(k)}(t-1), \quad y^{(k)}(t-1)\\ &\times x^{(k)}(t-2), \quad y^{(k)}(t-2) \times x^{(k)}(t-1), y^{(k)}(t-2)\\ &\times x^{(k)}(t-2), \quad x^{(k)}(t-1) \times x^{(k)}(t-2), \quad \boldsymbol{\theta}^{(K)}]\end{aligned} \tag{7}$$

In nonlinear dynamic systems, the number of candidate features M is usually very large due to the consideration of time dependencies and nonlinear characteristics. However, not all these features are necessarily useful, and not all of them should be included in the neural network. Training a neural network with an excessive number of inputs can be time-consuming. Identifying the essential features becomes challenging when all available features are utilized. Moreover, incorporating too many features may lead to overfitting, thereby reducing the model's overall robustness. To tackle these challenges, feature selection approaches are developed to identify the key features from a pool of candidates. Typically, the conventional feature selection method involves several steps. Following our previous work, we briefly present the procedures of feature selection as follows.

Firstly, the contribution of features of all the sub-datasets is calculated using various measures. Examples of such measures include Error Reduction Ratio (ERR), mutual information [19]. Let us denote this measure as $g(\bullet)$. Then, the index of the features with highest value of $g(\boldsymbol{\varphi}_m^{(k)})$ can be identified as

$$l_1 = \max_{m=1,2,\ldots M}(g(\boldsymbol{\varphi}_m^{(k)})). \tag{8}$$

Subsequently, the selected feature is excluded from the following selection process and the index of other key features can be identified as

$$l_i = \max_{m\in\{1,2,\ldots M\}, m\notin\{l_1,\ldots l_{i-1}\}}(g(\boldsymbol{\varphi}_m^{(k)})). \tag{9}$$

Importantly, an orthogonalization process is applied to the remaining features to ensure that the information already captured by the selected features is not reconsidered in the subsequent selection process.

Finally, the selected features can be represented as $\boldsymbol{\varphi}_{l_1}, \boldsymbol{\varphi}_{l_2}, \ldots \boldsymbol{\varphi}_{l_M}$. To determine whether enough features have been identified, model selection criterion is introduced and utilized as a threshold to halt the selection procedure.

These steps are summarized in Algorithm 1.

Algorithm 1. Feature selection for individual model

1. input response variable $\boldsymbol{y}^{(k)}$, candidate features $\boldsymbol{\varphi}_m^{(k)}$ of the k-th subsets, with $m = 1,2,3, \ldots, M$
2. initialize functions for calculating contribution and threshold as $g(\cdot)$ and $t(\cdot)$
3. assign $t(0) \leftarrow 0$
4. initialize $s \leftarrow 1$
5. while $t(s) < t(s-1)$ do
6. for $m = 1,2, \ldots, M$ do
7. compute $g(\boldsymbol{\varphi}_m^{(k)})$
8. identify $l_s \leftarrow arg \max_{1\le j\le M, j\notin l}\{g(\boldsymbol{\varphi}_j^{(k)})\}$
9. compute $t(s)$
10. update $s \leftarrow s+1$
11. end while
12. update selected features as $\boldsymbol{\varphi}_l{}^{(k)} \leftarrow [\boldsymbol{\varphi}_{l_1}{}^{(k)}, \boldsymbol{\varphi}_{l_2}{}^{(k)}, \ldots, \boldsymbol{\varphi}_{l_n}{}^{(k)}]$
13. output $\boldsymbol{\varphi}_l{}^{(k)}$

The above conventional methods are designed primarily for systems with only one output. For modelling tasks with multiple datasets, where multiple output variables are present, and each output is associated with its own set of features, these conventional methods may not be ideal. The key features selected for one output may not necessarily be effective for another output. Therefore, it is necessary to develop a method that can identify key features shared across all output variables.

3. The common structure neural network

In this section, we present the proposed method. First, we explain the basic concept. Next, we outline the algorithm for identifying shared features. Finally, we give an example of model design.

3.1. Basic idea

Consider a modelling task with multiple datasets, the input and output variables of each sub-dataset are described as:

$$\boldsymbol{x}^{(k)} = \begin{bmatrix} x_r^{(k)}(1) & x_r^{(k)}(1) & \cdots & x_r^{(k)}(1)\\ x_r^{(k)}(2) & x_r^{(k)}(2) & & x_r^{(k)}(2)\\ \vdots & & \ddots & \vdots\\ x_r^{(k)}(N) & x_r^{(k)}(N) & \cdots & x_r^{(k)}(N)\end{bmatrix} \tag{10}$$

$$\boldsymbol{y}^{(k)} = \begin{bmatrix} y^{(k)}(1)\\ y^{(k)}(2)\\ \vdots\\ y^{(k)}(N)\end{bmatrix} \tag{11}$$

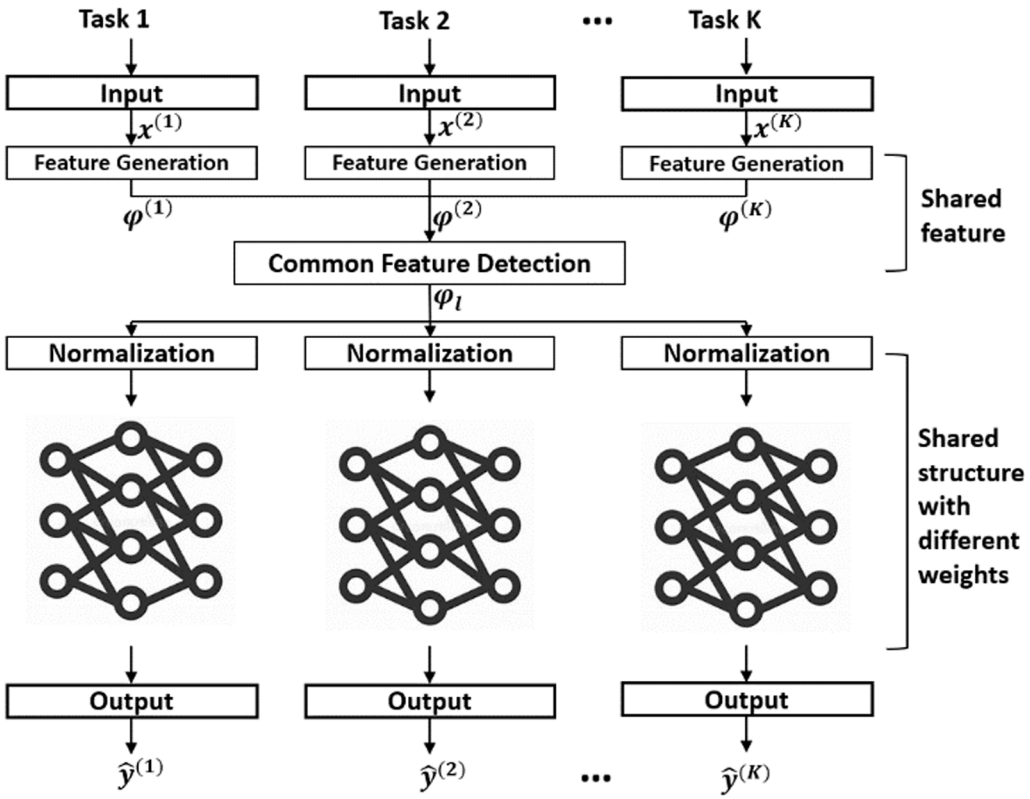

Fig. 1. Proposed model architecture. All networks have the same structure but are trained separately, leading to different parameters. The number of inputs for each network, φ_l, is much smaller than the original inputs to improve training efficiency.

where k denotes the index of a sub-dataset, with K representing the total number of sub-datasets. Each sub-dataset has the same number of variables and data samples. Specifically, r refers to the index of variables (with R being the total number of variables), and n refers to the index of data samples (with N as the total number of samples). Therefore, the input matrix for each sub-dataset has dimensions R by N.

The objective of the modelling task is to predict the output $y^{(k)}(t)$ from the inputs $x_r^{(k)}(t)$. Traditionally, to predict multiple outputs, a separate model is built for each output. For the k-th output, an individual model is built as:

$$\boldsymbol{y}^{(k)}(t) = f^{(k)}[\boldsymbol{x}^{(k)}(t), \quad \boldsymbol{\theta}^{(K)}] \tag{12}$$

where $\boldsymbol{\theta}^{(K)}$ represents the weights or parameters of the model and $f^{(k)}$ represents predicted model. When there are K outputs to predict, this traditional approach involves building a separate individual model for each output, denoted as $f^{(1)}, f^{(2)}, \ldots, f^{(K)}$.

The conventional process of constructing these individual models follows the typical data-driven modelling pattern, involving data preprocessing, creating training and test datasets, training the model, and validating its performance. For these individual models, there is no definitive answer to which model structure is superior, as their performance heavily relies on the characteristics of the target system and the quality of the data [33].

However, these individual models often share common dynamics and structures in certain scenarios. For example, when predicting pollution across six locations within a large area, pollutants from a single industrial site can impact all locations, and the weather conditions of one location affect the pollution of another location. Additionally, when there are many target outputs, the training cost becomes significantly high.

In these situations, it becomes essential to develop a common model structure that identifies shared features across all outputs for several key reasons: 1). Analyzing shared information across tasks can improve prediction accuracy compared to using separate models, as feature reduction helps prevent overfitting and makes the model more robust. 2). Model complexity and training costs are significantly reduced since only one model structure is needed instead of multiple models.3) Enhanced model interpretability makes it easier to understand the relationships between inputs and outputs.

This task is to identify a model structure, f, which can be shared by all the individual models $f^{(1)}, f^{(2)}, \ldots, f^{(K)}$. With this shared model structure capable of accommodating all subsets, the predictive models for each of the K outputs can be streamlined as follows:

$$y^{(k)}(t) = f\ [\boldsymbol{x}^{(k)}(t), \quad \boldsymbol{\theta}^{(K)}] \tag{13}$$

This approach allows the creation of only one unified model structure f for all K subsets. The weights or parameters of each model are deliberately designed to differ, accommodating the disparities among the K sub-systems. This not only facilitates the identification of common features, offering insights into the correlations among the subsets, but also enables the modeling of differences within the subsets, ensuring accurate predictions. A visual representation of the basic design is shown in Fig. 1.

3.2. Shared feature identification

As discussed above, the basic idea of the proposed model is to use a shared feature layer to select the key features for all the subsets. To achieve this, we propose a new measure that can calculate the average error reduction of all the subsets when adding a specific feature to the model. The new measure is named as average error reduction (AER), which can be calculated as

$$h_m^{(k)} = \sum_{k=1}^{K} [\beta_m^{(k)} \boldsymbol{\varphi}_m^{(k)}] \Big/ K. \tag{14}$$

Here, the weight factor is calculated as

$$\beta_m^{(k)} = (\boldsymbol{y}^{(k)} \boldsymbol{\varphi}_m^{(k)}) / (\boldsymbol{\varphi}_m^{(k)} \boldsymbol{\varphi}_m^{(k)}) \tag{15}$$

In these equations, the output associated with the k-th set of features, $\boldsymbol{y}^{(k)}$, and the inner product of the m-th feature with itself within the k-th set, $\boldsymbol{\varphi}_m^{(k)} \boldsymbol{\varphi}_m^{(k)}$, are considered. The Average Error Reduction (AER) extends the Error Reduction Ratio (ERR) used in single-output systems [19] to multi-output settings. It quantifies the average improvement in model accuracy across all subsets when a candidate feature is included. Eq. (14) aggregates each feature's contribution across sub-datasets, while Eq. (15) normalizes these contributions to ensure comparability across heterogeneous scales. The matrix H in Algorithm 2 stores the AER values for all features and subsets. Shared features are identified by selecting those with the highest AER values, indicating the greatest average reduction in prediction error. This greedy selection procedure prioritizes features that consistently enhance performance across tasks, ensuring that the shared model captures meaningful common patterns while remaining compact and interpretable.

Subsequently, we integrate the measure into the feature selection procedure, replacing $g(\bullet)$. As a result, the contribution of M candidate terms in K subsets can be represented in a matrix denoted by H:

$$\mathrm{H} = \begin{bmatrix} h_1^{(1)} & h_1^{(2)} & \cdots & h_1^{(k)} \\ h_2^{(1)} & h_2^{(2)} & & h_2^{(k)} \\ \vdots & & \ddots & \vdots \\ h_M^{(1)} & h_M^{(2)} & \cdots & h_M^{(k)} \end{bmatrix} \tag{16}$$

The updated algorithm allows for the selection of shared features $\boldsymbol{\varphi}_{l_1}, \boldsymbol{\varphi}_{l_2}, \ldots, \boldsymbol{\varphi}_{l_n}$ across all K subsets, as outlined in Algorithm 2. Through the evaluation of the AER for each feature, we can identify the most influential features that substantially contribute to reducing prediction errors across all sets. This enhanced shared feature selection process results in a more effective and robust feature identification.

It should be noted that the degree of nonlinearity in the feature dictionary was restricted to a maximum of degree 2 to balance representational flexibility and numerical stability. Higher-order polynomial terms were initially evaluated during preliminary experiments but were found to provide negligible improvement in predictive accuracy for most cases [8-11] while substantially increasing the number of candidate terms and the risk of overfitting. Since feature selection procedures already capture key nonlinear interactions, extending to higher degrees introduced redundant, weakly correlated terms that increased computational cost without improving generalization. Therefore, degree 2 provides an optimal compromise between model expressiveness, interpretability, and computational efficiency.

Algorithm 2. Shared feature selection for the CSNN model

1. input response variable $\boldsymbol{y}^{(k)}$, candidate features $\boldsymbol{\varphi}_m^{(k)}$ of all subsets, with $m = 1,2,3,\ldots,M$, and k= $1,2,3,\ldots,K$
2. initialize functions for calculating contribution and threshold as h(·) and $t(\cdot)$
3. assign $t(0) \leftarrow 0$
4. initialize $s \leftarrow 1$
5. while $t(s) < t(s-1)$ do
6. for $m = 1,2,\ldots,M$ and $k = 1,2,\ldots,$K do
7. compute $h_m^{(k)} \leftarrow \sum_{k=1}^{K} [(\boldsymbol{y}^{(k)} \boldsymbol{\varphi}_m^{(k)}) / (\boldsymbol{\varphi}_m^{(k)} \boldsymbol{\varphi}_m^{(k)}) \quad \boldsymbol{\varphi}_m^{(k)}] / K$
8. compute $\mathrm{H} \leftarrow \begin{bmatrix} h_1^{(1)} & h_1^{(2)} & \cdots & h_1^{(k)} \\ h_2^{(1)} & h_2^{(2)} & & h_2^{(k)} \\ \vdots & & \ddots & \vdots \\ h_M^{(1)} & h_M^{(2)} & \ldots & h_M^{(k)} \end{bmatrix}$
9. identify $l_s \leftarrow arg \max_{1 \le j \le M, j \notin l} (\mathrm{H})$
10. compute $t(s)$
11. update $s \leftarrow s + 1$
12. end while
13. update selected features as $\boldsymbol{\varphi}_l \leftarrow [\boldsymbol{\varphi}_{l_1}, \boldsymbol{\varphi}_{l_2}, \ldots, \boldsymbol{\varphi}_{l_n}]$
14. output $\boldsymbol{\varphi}_l$

The shared features selected through the feature selection process are used to build the neural network structure (Section 3.3). This model is designed as a shared structure capable of accommodating all the subsets and generating predictions for the related outputs. The fundamental idea of this approach is that the model, constructed using the identified shared features, effectively captures the common dynamics present in all the individual groups.

Note that the weights of each individual neural network within the shared structure are separately estimated using the individual subsets. As a result, this approach enables the model to adequately represent both the shared characteristics among all different subjects and the distinct characteristics of each individual group.

This design has several key benefits. Firstly, by analyzing time dependencies in historical data, the shared feature generation and selection layers reduce the complexity of neural networks, speeding up training and cutting costs by minimizing the number of model structures needed. Secondly, sharing information across tasks boosts prediction accuracy compared to using separate models for each task. The feature reduction process helps prevent overfitting, making the model more robust, especially for challenging tasks like predicting peak values. Lastly, the model offers insights into the common factors affecting all groups, improving understanding of the underlying relationships.

3.3. Shared model structure

The final step in the modeling process is to customize the neural network structure for the specific task. The shared neural network can be adjusted to match the unique characteristics of the target systems. Fig. 2 illustrates the entire process of building the proposed model. In this example, we use air pollution prediction and demonstrate a basic feedforward neural network structure. Since time-lagged features were already generated and selected during the feature identification process in Section 3.2, a recurrent neural network is not necessary at this stage.

The neural network structure in Fig. 2 is driven by the need to capture temporal dependencies, enhance feature representation, and

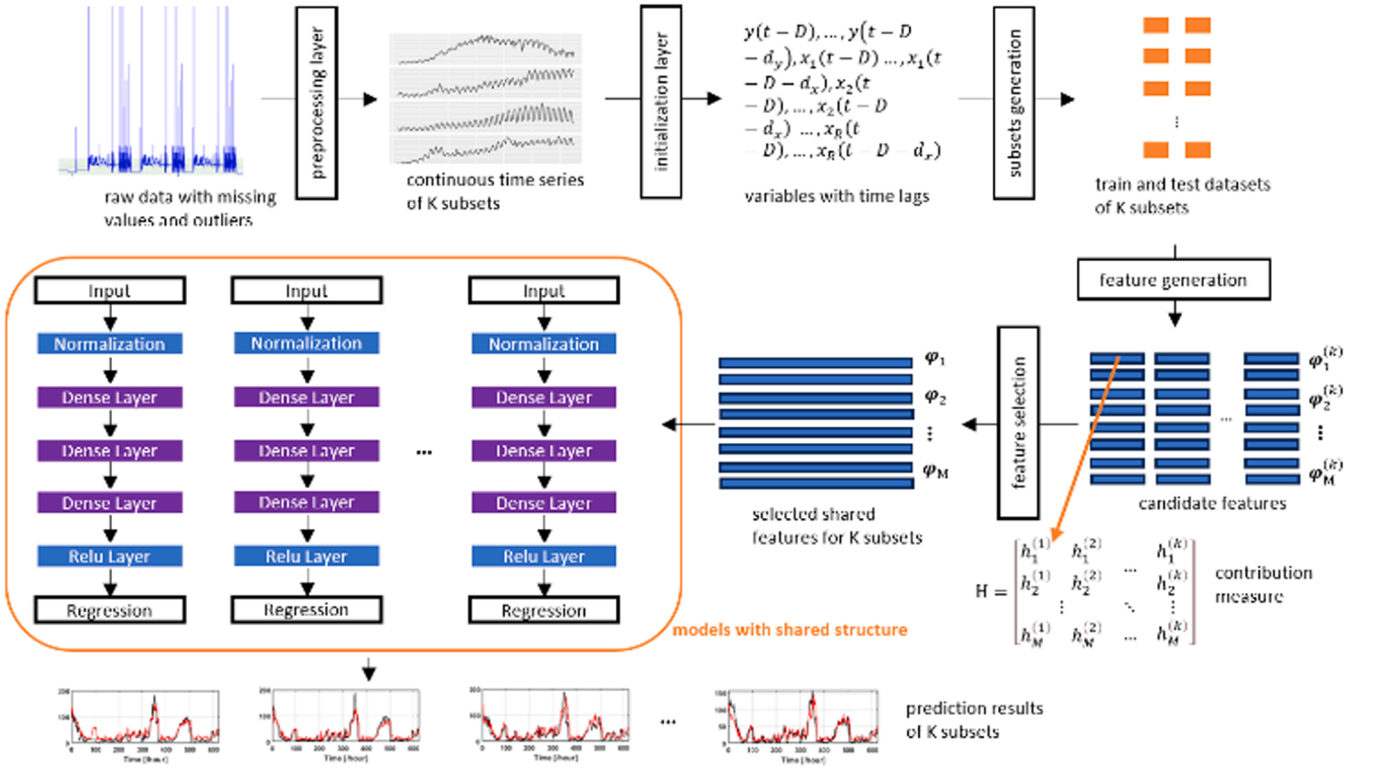

Fig. 2. The proposed CSNN model for predicting air pollution. The structure of the neural network can be tailored to accommodate various scenarios and datasets. This example illustrates its application in air pollution case studies.

optimize predictive performance in air pollution forecasting. The preprocessing and initialization layers address data integrity and time lag considerations, crucial for time-sensitive datasets like air pollution. Subset generation and feature derivation layers are crafted to systematically explore both linear and nonlinear relationships, ensuring that complex interactions are not overlooked. By selecting shared features across all subsets, the model generalizes well across different data partitions, enhancing robustness. The following layers were designed to balance interpretability, computational efficiency, and predictive performance across diverse tasks. We adopt a feedforward neural network with three hidden layers comprising 128, 64, and 8 neurons, respectively. These dimensions were selected based on empirical validation to provide sufficient model capacity for learning complex nonlinear interactions while avoiding overfitting. The dimensionality gradually reduces to promote feature compression and abstraction. We use ReLU (Rectified Linear Unit) as the activation function due to its computational simplicity, non-saturating gradient behavior, and compatibility with the non-negative nature of outputs like air pollution levels. Additionally, ReLU has shown consistent performance across time series prediction tasks in previous studies. The use of shared feature selection followed by individualized parameter learning enables the model to generalize well across tasks with similar underlying dynamics (e.g., different geographical locations or companies), while preserving task-specific ability through separate weight optimization.

Finally, the model performance is assessed using several metrics, such as the correlation coefficient, R^2, and Root Mean Square Error (RMSE), as follows:

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y}_i)^2} \tag{17}$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{18}$$

Additionally, we validate the model by examining the importance of the selected features and reviewing supporting evidence in the literature.

Our experimental results show that the proposed framework can approximate various target systems, providing both accurate predictions and reduced model complexity. While the model discussed above is tailored to a specific use case, the proposed architecture is versatile and can be adapted to many different scenarios. In the following sections, we will present three case studies that demonstrate how this architecture can improve prediction performance and model interpretability across various applications.

3.4. Convergence of CSNN

The training of CSNN consists of two sequential and independent stages: (1) deterministic feature selection by Algorithm 2 (OFR-based selection), and (2) standard neural-network optimization for each subtask network trained on the selected inputs.

For the first stage, Algorithm 2 implements Orthogonal Forward Regression (OFR) to determine the shared feature set S. Let $\Phi = [\phi_1, \ldots, \phi_M]$ denote the full candidate feature matrix and $r^{(t)}$ the residual vector after t selections.

At each step t, Algorithm 2 selects

$$m_t = \underset{m}{\operatorname{argmax}}\,(\mathrm{AER}_m)$$

$$= \underset{m}{\operatorname{argmax}}\left(\frac{1}{K}\sum_{k=1}^{K}\frac{\langle r_k^{(t)}, \phi_m^{(k)}\rangle^2}{\left\|r_k^{(t)}\right\|_2^2 \left\|\phi_m^{(k)}\right\|_2^2}\right) \tag{19}$$

then orthogonalizes ϕ_{m_t} against the previously selected features (lines 9–11 in Algorithm 2) and updates the residuals as:

$$r_k^{(t+1)} = r_k^{(t)} - \beta_{m_t}^{(k)} \phi_{m_t}^{(k)} \tag{20}$$

Because each newly chosen feature is orthogonal to the previously selected span, we have the deterministic, monotone relation

$$\left\| r_k^{(t+1)} \right\|_2^2 = \left\| r_k^{(t)} \right\|_2^2 - \left\| \beta_{m_t}^{(k)} \phi_{m_t}^{(k)} \right\|_2^2 \tag{21}$$

and

$$\left\| r_k^{(t+1)} \right\|_2^2 \le \left\| r_k^{(t)} \right\|_2^2 \tag{22}$$

with equality only when all remaining features are orthogonal to the residual.

Hence, the empirical error decreases (or stays constant) at each iteration, and since at most M features exist, Algorithm 2 terminates in a finite number of deterministic steps with a fixed selected set $S = \{m_1, \ldots, m_s\}$. No stochastic element is involved, so the feature-selection process itself converges deterministically.

For the second stage, once S is fixed, every task k uses only the selected inputs $\tilde{x} = P_S x \in \mathbb{R}^s$, where P_S extracts the coordinates indexed by S. Each subtask network $f_k(\tilde{x}; \theta_k)$ is trained to minimize the empirical loss

$$\widehat{\mathscr{L}}_k(\theta_k) = \frac{1}{N_k} \sum_{i=1}^{N_k} \ell\left(f_k(\tilde{x}_i^{(k)}; \theta_k), y_i^{(k)}\right), \quad \tilde{x}_i^{(k)} \tag{23}$$

The objective has the same functional form as the standard neural-network optimization on the original inputs x, differing only in dimensionality from M to s. Therefore, standard convergence results for stochastic-gradient methods apply directly. Assuming the usual conditions-L-smooth loss, bounded variance of stochastic gradients, and step size $\eta_t \le 1/L$ the expected decrease of the empirical loss satisfies

$$\mathbb{E}\left[\widehat{\mathscr{L}}_k(\theta_{k,t+1}) - \widehat{\mathscr{L}}_k(\theta_{k,t})\right]$$

$$\le -\frac{\eta_t}{2} \mathbb{E}\left[\left\| \nabla_{\theta_k} \widehat{\mathscr{L}}_k(\theta_{k,t}) \right\|_2^2\right] + \frac{L\eta_t^2 \sigma^2}{2} \tag{24}$$

which leads to the standard non-convex convergence guarantee:

$$\frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\left[\left\| \nabla_{\theta_k} \widehat{\mathscr{L}}_k(\theta_{k,t}) \right\|_2^2\right] = \mathscr{O}\left(\frac{1}{\sqrt{T}}\right) \tag{25}$$

Because the mapping $x \mapsto P_S x$ is linear and deterministic, replacing x by $\tilde{x}$ does not alter the structure of the optimization problem or the assumptions required for convergence-only the constants (e.g., Lipschitz and gradient-variance terms) change, typically becoming smaller as the input dimension $s < M$.

In summary, by selecting a small number of shared features across all tasks before training, CSNN reduces the dimensionality of the input space, which simplifies the optimization landscape and typically leads to faster and more stable convergence in practice [13]. From a multi-task learning perspective, constraining all subtask networks to operate on the same reduced input space S acts as an implicit regularizer [23]. By restricting the hypothesis space to functions defined over S rather than the full feature set, the effective model capacity is reduced in a principled way, leading to tighter generalization bounds compared to independently trained models [34],40]. Intuitively, features selected because they consistently reduce prediction error across all tasks are less likely to reflect task specific noise, thereby improving generalization to unseen data.

4. Empirical evaluation and domain applications

To evaluate the effectiveness of the proposed framework, three case studies are presented in this section. The first is a simulation study designed to validate the core mechanism of shared feature identification under varying noise conditions. The second applies the framework to a real-world air pollution forecasting task across six spatially distributed monitoring stations, demonstrating the model's ability to capture spatial dependencies. The third extends the framework to a financial domain, applying it to stock price prediction across three retail companies. Across all three scenarios, the proposed method is evaluated against multiple baseline models in terms of predictive accuracy, training efficiency, and interpretability.

4.1. Simulation

We create a simulation dataset that serves as an illustrative example. We examine a scenario involving a system with a single input and multiple outputs. In this context, the input sequence is denoted as 'x', while the output sequences are denoted as a set $y^{(k)}$ for $k = 1, 2, .., 6$. The subsequent guidelines are employed during the generation of both the input and output time series. The input sequence is produced using a random vector comprising 100 sampled values ranging from −1–1, with precision to two decimal places. The output sequences are generated as a 100 by 6 matrix, signifying the existence of six distinct output sequences, each consisting of 100 sampled values. To establish the correlation between the input and output, a nonlinear dynamic system is employed:

$$y^{(k)}(t) = a^{(k)} \times \sqrt{|y^{(k)}(t-1)|} \times x(t-1) + b^{(k)} \times x(t-1)^2 + c^{(k)} \times x(t-1) \times x(t-2) + d^{(k)} \times x(t-2) \times y^{(k)}(t-2) + \xi(t) \tag{26}$$

In this system, the time-lagged sequences $y^{(k)}(t-1)$, $y^{(k)}(t-2), x(t-1), x(t-2)$ are incorporated, with $a^{(k)}, b^{(k)}, c^{(k)}$, $d^{(k)}$ representing the parameters for $k = 1, 2, .., 6$. While the model

Table 1
Performance comparison of the CSNN model and individual models on the simulation dataset. The CSNN model (Common Structure Neural Network) uses identical shared features across all six subsets, whereas the individual models use separate features for each of the six subsets. The experiments are conducted under three different noise levels. The performances of the models are evaluated using Correlation Coefficient (CC) and Root Mean Square Error (RMSE).

Noise	Subset	CSNN Model		Individual Model	
		CC	RMSE	CC	RMSE
$\xi(t) \sim N(0, 5)$	$y^{(1)}$	0.4701	4.8332	0.4612	4.9443
	$y^{(2)}$	0.2217	5.0123	0.2270	5.0648
	$y^{(3)}$	0.2107	5.2063	0.1766	5.3391
	$y^{(4)}$	0.3192	4.9205	0.3022	4.9956
	$y^{(5)}$	0.3593	4.8201	0.3441	4.9548
	$y^{(6)}$	0.5211	4.8937	0.4954	5.2322
	Average	0.3503	4.9477	0.3344	5.0885
$\xi(t) \sim N(0, 1)$	$y^{(1)}$	0.2763	1.0145	0.2590	1.0203
	$y^{(2)}$	0.3658	1.0127	0.3439	1.0230
	$y^{(3)}$	0.5863	1.0407	0.5727	1.0548
	$y^{(4)}$	0.5251	1.0263	0.5080	1.0405
	$y^{(5)}$	0.5044	1.0257	0.4977	1.0303
	$y^{(6)}$	0.5277	1.0457	0.5151	1.0539
	Average	0.4643	1.0276	0.4494	1.0371
$\xi(t) \sim N(0, 0.1)$	$y^{(1)}$	0.7713	0.3228	0.7698	0.3242
	$y^{(2)}$	0.4937	0.3294	0.4844	0.3316
	$y^{(3)}$	0.6728	0.3504	0.6762	0.3491
	$y^{(4)}$	0.3413	0.3184	0.3152	0.3222
	$y^{(5)}$	0.8117	0.3579	0.8084	0.3625
	$y^{(6)}$	0.7874	0.3338	0.7798	0.3394
	Average	0.6464	0.3354	0.6390	0.3382

components $x(t-1)^2$, $x(t-1)\times x(t-2)$, and $x(t-2)\times y^{(k)}(t-2)$ can typically be accommodated using straightforward nonlinear terms in a regression model, the term $\sqrt{|y^{(k)}(t-1)|}\times x(t-1)$ introduces complexity beyond interactions of linear terms. This type of term often needs a more complex model structure to fit, such as higher-degree nonlinear terms or a deep neural network architecture.

From the system design, the six sub-systems have a consistent structure but differ in parameters. This similarity provides an ideal foundation for testing the model. Furthermore, to simulate real-world situations where data collection processes are influenced by disruptions, a substantial noise sequence $\xi(t)$ is incorporated into the system. Three noise levels were tested to evaluate the sensitivity and robustness of the proposed method. This inclusion addresses the difficulties posed by real-world scenarios and highlights the constraints of human-designed models in accurately capturing complex nonlinear dynamics.

For this simulated dataset, we use two methods to build predictive models. First, we use a conventional approach to create separate models for each output. Next, we apply the proposed approach to build a single model structure that predicts all outputs simultaneously. Both methods use the same procedures for generating training and testing datasets, training the neural network, and validating the model, ensuring a fair comparison between the shared features and the traditional approach. For this simple simulation, the neural networks will have a hidden feedforward layer with 64 neurons. The hyperparameters set the neural network training to run for up to 100 epochs with mini batches of 64 samples, using an initial learning rate of 0.01.

The model performances are summarized in Table 1. We applied three noise levels, represented as white noise with zero mean and variances of 0.1, 1, and 5, respectively. The results show that the common structure model performs well and slightly outperforms individual models under lower noise conditions (variances of 0.1 and 1). Since the method is intended for modelling across multiple sub-datasets in high-uncertainty settings, our analysis mainly focuses on the results under the highest noise level (variance of 5). The results indicate that the proposed models outperform traditional methods, showing an average performance improvement of 4.75% in the correlation coefficient. Additionally, the advantages in individual outputs are notable, (e.g., with a 5.19% higher correlation coefficient for the 6th output). A visual comparison between the performances of two sets of models is presented in Fig. 3. These results demonstrate enhancements in correlation coefficients, prediction efficiency, and error reduction achieved by the proposed model.

Table 2
Derived common features for the simulation dataset. The features are fully interpretable with *x(t−1) × x(t−2)* represents the interaction between the input measured one time unit ago and the input measured two time units ago.

Index	Feature	AERs
1	$x(t-2)\times y(t-2)$	1.8929
2	$x(t-1)$	1.8711
3	$x(t-1)\times x(t-2)$	1.8560
4	$x(t-1)\times x(t-1)$	1.8467
5	$x(t-2)$	1.8397
6	$x(t-1)\times y(t-1)$	1.8338
7	$x(t-3)\times y(t-2)$	1.8314
8	$y(t-3)\times y(t-3)$	1.8293

Furthermore, Table 2 summarizes the identified shared features, alongside associated t-values and AERs used for validation. It is noticeable that a significant portion of nonlinear terms, such as $x(t-1)^2$, $x(t-1)\times x(t-2)$, and $x(t-2)\times y^{(k)}(t-2)$, are accurately identified. The term $\sqrt{|y^{(k)}(t-1)|}\times x(t-1)$is represented through supplementary nonlinear features, subsequently refined by the neural network in the subsequent stage to provide a more precise depiction of the $\sqrt{|y^{(k)}(t-1)|}\times x(t-1)$ term.

To further evaluate model robustness, we extended the simulation experiment to test the CSNN framework under missing-data and out-of-distribution (OOD) conditions. For the missing-data test, we randomly masked 5%, 10%, and 20% of input features (missing completely at random) and applied simple linear interpolation along the temporal dimension for imputation. For the OOD test, we introduced a controlled mean shift of + 10% and + 20% to the first three input variables to simulate moderate distributional drift. In both cases, the trained common DNN model was evaluated without retraining using the same simulation test set. The baseline model achieved an RMSE of 0.3394 and a correlation coefficient (CC) of 0.7798. Under 5%, 10%, and 20% missingness, RMSE increased slightly to 0.3606, 0.3613, and 0.3782, while CC decreased modestly to 0.7476, 0.7460, and 0.7172, respectively. Under OOD shifts of + 10% and + 20%, RMSE remained nearly

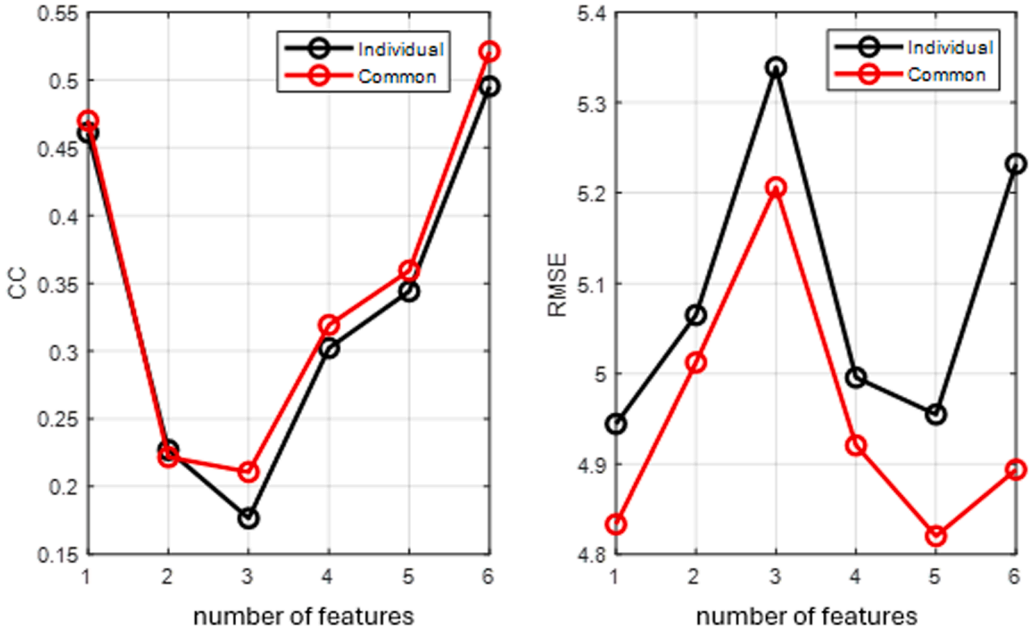

Fig. 3. Comparison of the performance between two sets of models on the simulation dataset. The red line represents the performance of models with common features, while the black line indicates the averaged performance of five individual models. The x-axis shows the index of all six subsets.

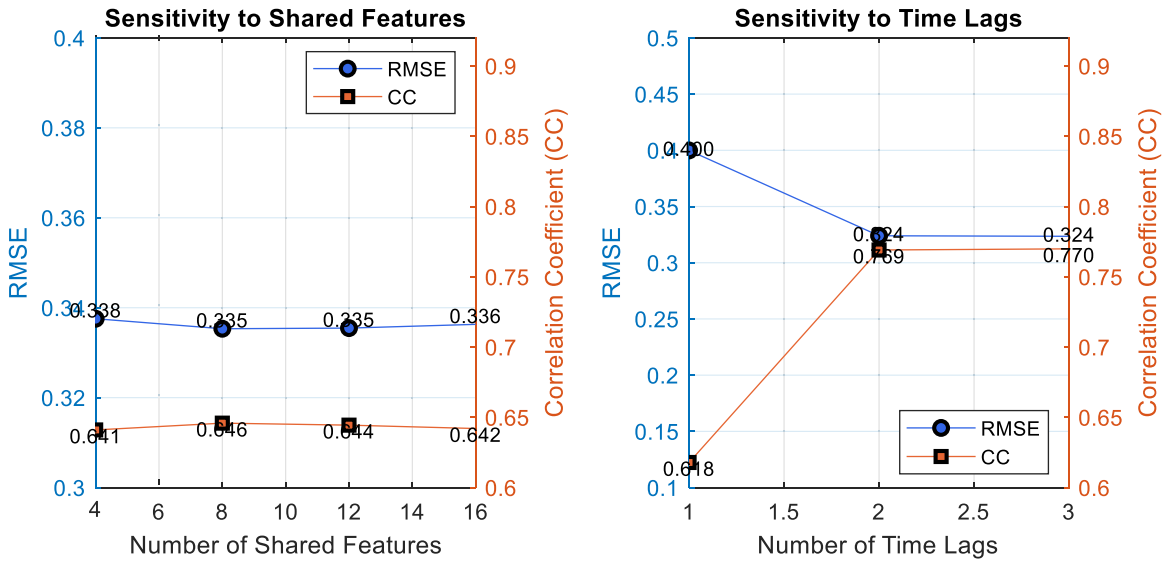

Fig. 4. Sensitivity of the CSNN model to the number of shared features and time lags. The blue line represents the RMSE of CSNN models with various number of features and time lags. The orange line indicates the Correlation Coefficient.

unchanged at 0.3412 and 0.3451, and CC stayed stable at 0.7777 and 0.7742. These results indicate that the CSNN framework degrades gracefully under incomplete inputs and maintains stable predictive performance even when feature distributions shift moderately, confirming their robustness and generalization capability in noisy and imperfect data environments.

Fig. 4 presents the sensitivity of the CSNN model to the number of shared features and time lags. The left panel shows how prediction accuracy varies with the number of shared features (4–16), and the right panel shows sensitivity to the number of time lags (1–3). As the number of shared features increases from 4 to 8, RMSE decreases from approximately 0.34–0.32, while the correlation coefficient (CC) increases from 0.75 to 0.78, after which performance stabilizes, indicating that moderate feature sharing achieves the best balance between model accuracy and complexity. Similarly, increasing the number of time lags from 1 to 2 substantially improves CC (from 0.68 to 0.77) and reduces RMSE (from 0.39 to 0.33), but further expansion to 3 lags yields marginal gains. These findings confirm that CSNN is not overly sensitive to hyperparameter choices and maintains strong predictive performance under a wide range of configurations, demonstrating good practical robustness.

It's important to note that simulation data cannot fully capture the complexities of real-world situations, and our main goal here is to demonstrate the basic idea of the proposed method. When used with real datasets, issues like predicting peak values can become more challenging. In these cases, the proposed model's performance improvement is more noticeable. In addition, the proposed approach only uses a single model structure and a set of shared features. This efficiency in real big data computation leads to overall better efficiency for the task.

Table 3
Pollution variables from the air pollution dataset. The variables O3-1h and O3-8h refer to the ozone (O3) measurements taken one hour and eight hours ago, respectively.

Guanyuan	Changping	Huairou	Dingling	Shunyi	Langfang
G-AQI	C-AQI	H-AQI	D -AQI	S -AQI	L-AQI
G-PM2.5	C-PM2.5	H-PM2.5	D -PM2.5	S-PM2.5	L-PM2.5
G-PM12	C-PM12	H-PM12	D -PM12	S-PM12	L-PM12
G-CO	C-CO	H-CO	D -CO	S-CO	L-CO
G-NO	C-NO	H-NO	D -NO	S-NO	L-NO
G-O3-1h	C-O3-1h	H-O3-1h	D -O3-1h	S-O3-1h	L-O3-1h
G-O3-8h	C-O3-8h	H-O3-8h	D -O3-8h	S-O3-8h	L-O3-8h
G-SO2	C-SO2	H-SO2	D -SO2	S-SO2	L-SO2

4.2. *Air pollution case study*

High exposure to air pollution is linked to various health problems. With industrialization and economic growth, pollution has become a major global concern, especially in developing countries. Predicting air pollution is crucial, and many data-driven models have been developed to forecast air quality based on local weather and pollutant data. However, creating a model that can predict pollution levels at multiple locations and understand their spatial relationships is challenging. To tackle this, the proposed model is designed to identify common factors affecting air quality across different areas and predict PM2.5 levels.

A Python API was developed to collect data from weather and pollution websites for six locations in the Beijing area. We used an SQL database to store the data. Table 3 lists the pollution variables collected. Locations are abbreviated as follows: Guanyuan (G), Changping (C), Huairou (H), Dingling (D), Shunyi (S), and Langfang (L). We also included data from a public weather database, adding variables such as temperature, rain, humidity, and wind speed.

Due to sensor failures, measurement errors, and data transmission issues, the raw dataset contains numerous missing values and outliers. To address these problems, standard pre-processing techniques were applied. Missing values were filled using interpolation methods. Outliers, defined as data points that deviate substantially from surrounding observations, were detected and removed through smoothing and thresholding approaches. For categorical variables such as season and wind direction, dummy variables were created to represent each category. These binary variables take a value of 0 or 1 to indicate the absence or presence of a particular class, ensuring that each category is represented equally and allowing for fair comparison of their contributions in the analysis. After pre-processing, the datasets are merged based on sampling timestamps. Fig. 5 shows the variables for one location, Guanyuan, demonstrating the successful removal of abnormal values and missing data points. This study focuses on building a multi-task model to predict PM2.5 levels at all six locations using historical weather and pollution data (as in Table I). We use 75% of the dataset for training and the remaining 25% for testing and evaluating the model's performance.

Initial settings are determined through trial experiments. For predicting six hours into the future, the model's time delay is set at $D = 6$, and the maximum time lags are $d_x = d_y = 2$ [13]. The shared feature selection layer identifies 10 key features. These features, related to AER

Table 4
Ablation and comparison of CSNN, related variants and other models. The second model, the Individual Deep Neural Network (I-DNN), adopts the same architecture as the CSNN model but excludes shared inputs. The third model, the Multi-Task Neural Network (MT-NN), utilizes shared features and includes a single dense layer with 16 neurons. The fourth model, the Individual Neural Network (I-NN), relies on original inputs (i.e., without shared features) and incorporates a single dense layer of 16 neurons. The fifth model, the Individual LSTM (I-LSTM), processes original inputs using an LSTM layer for feature selection rather than employing the proposed shared-feature approach. The sixth model, I-Attention, extends the I-LSTM by integrating an additional attention mechanism while continuing to operate on non-shared features.

Criterion	Model	G	C	H	D	S	L	A
Root	CSNN	20.18	16.15	17.08	17.34	20.91	20.31	18.66
Mean	I-DNN	21.43	18.79	23.21	27.97	24.84	21.83	23.01
Square	MT-NN	27.85	28.38	27.90	28.88	34.35	27.89	29.21
Error	I-NN	23.19	22.19	20.94	24.10	24.51	24.66	23.26
	I-LSTM	32.79	33.39	32.68	34.31	38.08	32.35	33.94
	I-Attention	24.96	32.72	18.19	24.36	34.95	22.93	26.35
Correlation	CSNN	0.795	0.869	0.844	0.866	0.809	0.789	0.829
Coefficient	I-DNN	0.783	0.818	0.805	0.796	0.729	0.761	0.782
	MT-NN	0.805	0.830	0.812	0.832	0.785	0.784	0.808
	I-NN	0.768	0.805	0.782	0.778	0.752	0.753	0.773
	I-LSTM	0.094	0.095	0.101	0.111	0.096	0.783	0.096
	I-Attention	0.762	0.855	0.847	0.839	0.761	0.784	0.808

values, are shown in Fig. 6 and are fully interpretable, reflecting the physical significance of the input variables. For example, D-PM2.5(t−6) represents the PM2.5 value in Dingling from six hours ago, and S-SO2 (t−6) × L-SO2(t−7) shows the interaction between Shunyi's SO2 level from six hours ago and Langfang's SO2 level from seven hours ago. These features highlight the spatial relationships between locations. The results indicate that Dingling and Langfang are significant factors. Emissions of NO, O3, and SO2 from Dingling, and NO and SO from Langfang, notably increase pollution. Shunyi also affects air quality in other regions due to its SO2 emissions. The historical PM2.5 levels of Guanyuan and Huairou also influence air pollution, while Changping has minimal impact.

The shared features identified by the CSNN, such as lagged PM2.5, SO_2, and NO values across nearby sites, are consistent with known environmental mechanisms reported in air pollution research. Previous studies have shown that spatial interactions among monitoring stations and regional transport play a key role in PM2.5 formation and prediction accuracy [12,22], while secondary pollutants such as SO_2 and NO_x are recognized precursors of fine particulate matter and regional haze formation [2,35]. This agreement with established atmospheric knowledge suggests that the shared features selected by the model reflect meaningful physical processes.

The features are integrated into a neural network consisting of several components (as presented in Section 3.3). It has a normalization layer for standardizing input time series, three dense layers with 128, 64, and 8 neurons to process the data, a ReLU layer to handle the non-negative nature of air pollution predictions, and a regression layer to produce the predicted time series for all subsets. The hyperparameters set the neural network training to run for up to 100 epochs with mini batches of 64 samples, using an initial learning rate of 0.01.

Prediction accuracy is evaluated using root mean squared error (RMSE) and correlation coefficient (CC), as detailed in Table 6. The results show that the proposed model delivers notably accurate predictions, especially for PM2.5 levels in Changping and Dingling and performs consistently well across other locations. Fig. 7 visually compares the predicted PM2.5 values against observed values for all six locations, highlighting the model's ability to accurately predict most PM2.5 values, including peaks (PM2.5 $>$ 100). However, the model does struggle to predict peak values for Shunyi and Dingling, which is expected due to the imbalance between the large amount of training data from normal periods and the smaller amount from peak periods. Despite this, the model consistently achieves high prediction performance.

Table 4 compares the performance of six models across six locations using Root Mean Square Error (RMSE) and Correlation Coefficient (CC) as evaluation metrics. The results show that the CSNN model, which employs shared features, consistently achieves the lowest RMSE and highest CC, indicating superior predictive accuracy and robustness. In contrast, individual models without shared features such as I-DNN, I-NN, I-LSTM, and I-Attention generally perform worse, with I-LSTM and I-Attention yielding the highest errors despite their more complex architectures. The I-Attention model adds an attention layer to the individual LSTM to help the network focus on the most relevant parts of the input. While attention mechanisms are often used to enhance model interpretability and performance, the I-Attention model shows relatively lower accuracy, similar to I-LSTM. This suggests that the use of attention alone is not sufficient to improve prediction accuracy when shared features are not used. The multi-task neural network (MT-NN), which also uses shared representations, performs better than individual models but remains slightly inferior to CSNN. These findings highlight the effectiveness of shared feature learning in enhancing generalization across spatial domains and suggest that added complexity from recurrent or attention mechanisms may not yield performance gains in the absence of shared inputs.

Fig. 7 provides a visual comparison of the CSNN and I-DNN model predictions versus actual PM2.5 values. It shows that while both models predict low PM2.5 values well, the CSNN model aligns more closely with the reference line, indicating smaller prediction errors. The CSNN model significantly outperforms the other models in predicting PM2.5 levels above 100. Specifically, the prediction errors of other models, especially for Guanyuan and Dingling, are notably higher. The scatter plot shows that the CSNN model excels in predicting PM2.5 values both during low periods and peak times. In terms of training time on the same PC setup, the CSNN model takes about 6 s, while the I-NN model takes 2 min, and the I-LSTM model takes 7 min. This indicates that the CSNN model is much more efficient compared to the others.

Table 4 can be interpreted as an ablation of the main components of the CSNN architecture. The I-DNN variant removes the shared feature selection module, isolating the contribution of common input features across tasks. The MT-NN model retains feature sharing but removes task-specific heads, allowing us to assess the effect of individualized adaptation. The LSTM-based and attention-based variants substitute the AER-driven feature sharing with sequential or attention mechanisms, respectively. The consistent superiority of the CSNN across all configurations confirms that both the shared feature selection and task-specific refinement jointly contribute to predictive performance.

To further evaluate interpretability, we applied the Local Interpretable Model-agnostic Explanations (LIME) method to the I-Attention model. This model was selected because it achieved the second-best predictive performance and inherently incorporates an attention mechanism that provides built-in interpretability. For each of the six prediction subtasks, we randomly selected 60 test instances and generated 3000 perturbed samples around each instance using Gaussian noise

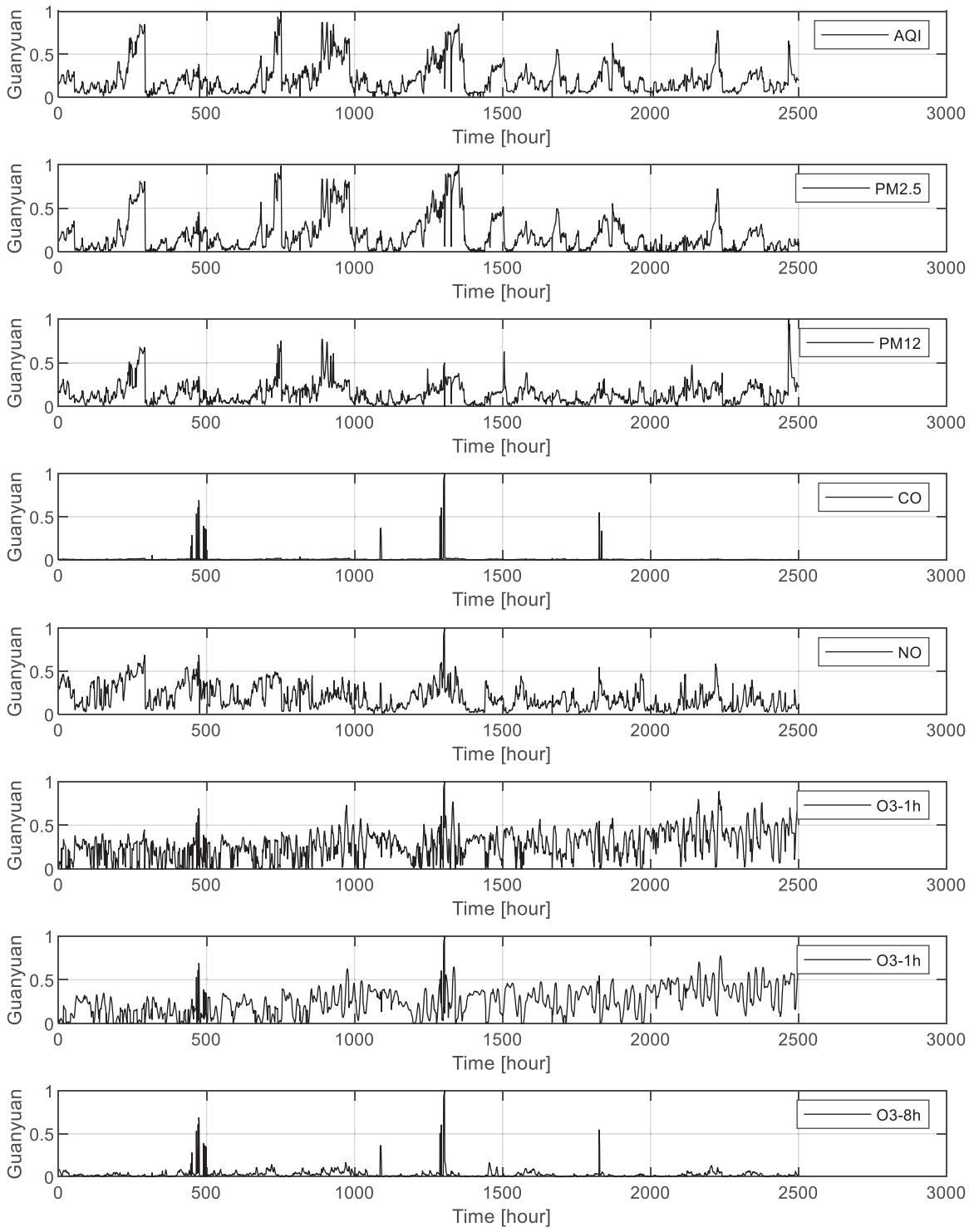

Fig. 5. Time series of the measured variables in Guanyuan. Note that this data represents one subset out of the six. The time series for the other subsets follows a similar pattern, though the values may differ.

scaled by the feature-specific standard deviation. A kernel width of 0.75 controlled locality weighting, and each explanation was repeated five times to assess stability across bootstrap runs. Fidelity was measured by the R-squared value between the surrogate and model predictions, while stability was quantified by the average Spearman correlation of feature rankings across bootstraps. The results show that the I-Attention model achieved fidelity values of -6.8, -7.5, -8.1, -9.3, -10.7, -16.2 and stability values of 0.754, 0.755, 0.757, 0.760, 0.761, 0.770 across the six prediction tasks. On average, the model achieved a fidelity of -9.1 and a stability of 0.76. Although the local linear surrogates have limited fit due to the model's nonlinear structure, the consistently high stability values indicate that LIME repeatedly identifies the same key input features across runs. These results confirm that the I-Attention model produces robust and consistent local explanations, complementing the global interpretability offered by the CSNN's shared feature selection mechanism.

While the CSNN provides built-in interpretability by identifying shared input features that contribute to prediction across multiple sub-

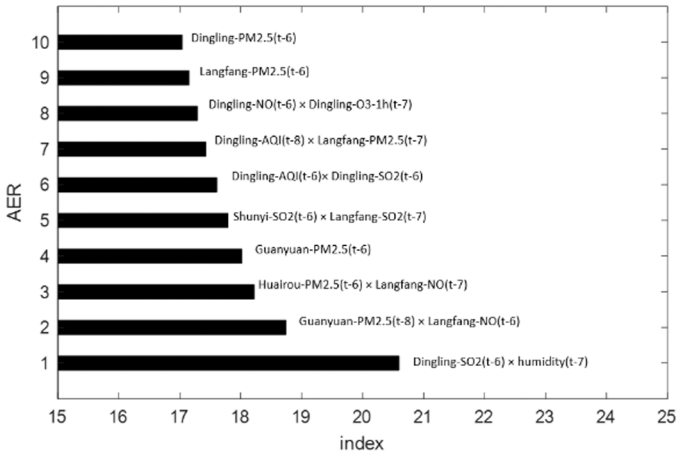

Fig. 6. Shared features and AER values. High AER values indicate a significant reduction in prediction error for air pollution across the six locations when the feature is included in the model. These ten features will be utilized to establish a common structure for predicting pollution levels at the six locations.

datasets, it differs conceptually from post-hoc explanation tools such as LIME and SHAP (SHapley Additive exPlanations). LIME and SHAP are widely used to interpret complex black-box models by approximating local model behavior or attributing prediction outcomes to individual input features. In contrast, CSNN's interpretability is embedded directly in its design through the shared feature selection layer, which highlights input variables that consistently influence predictions across all tasks. This enables global, domain-relevant insights into system dynamics rather than localized, instance-specific approximations. Consequently, CSNN supports transparent model construction and a more principled understanding of underlying relationships without relying on an external interpretability layer.

4.3. Stock price case study

In fact, the proposed model is also capable of numerical analysis of social science and economics problems. These often involve non-physical aspects like stock prices and company performance. Investigating correlations between these factors is essential, as it can provide insights into market characteristics through data-driven approaches, which have become more crucial due to the increased size and complexity of data in recent years. In this case study, we used the proposed method to estimate stock prices using a single model structure, enabling us to analyze common features across various stock markets.

Daily price of U.S. stocks and macroeconomic indicators are the data sets of this experiment. To be specific, we collect daily stock price of retail companies that are listed on NYSE, NYSE American and NASDAQ within the time frame of January 3, 2000, to December 30, 2022, from Capital IQ [36]. This 23-year window period encompasses diverse market scenarios, including economic upturns and economic downturns (e.g., financial crisis, COVID pandemic, etc.), enhancing the potential for more accurate predictions. Within this sample period (23 years), we also collect several macroeconomic variables that may have potential influence on the economy and market from the Federal Reserve Bank of St. Loui, including interest rate, exchange rate, corporate bond yield, oil price. The stock price data set offers basic details on stock trading, encompassing the daily closing price, share trading volume and market return. The closing price, which is used for the model prediction, is the last trading price of the stock during a given day, while the share trading volume indicates the total number of shares traded within a given day. Market return is the daily return of the value-weighted portfolio of all NYSE, NASDAQ companies. In our experiment, we have selected 3 companies from the retail sector that possess ample daily price data, namely Kroger Co, Weis Markets Inc, Village Super Market Inc.

The macroeconomic data set provides potential information for the prediction of stock price. We focus on 3-month, 6-month and 1-year yield of Treasury bill, yield on U.S. Treasury securities at 2-year, 3-year, 5-year, 10-year, 30-year constant maturity, Japanese Yen to U.S. dollar spot exchange rate, U.S. Dollars to U.K. Pound Sterling spot exchange rate, Chinese Yuan Renminbi to U.S. Dollar spot exchange rate, Canadian Dollars to U.S. Dollar spot exchange rate, oil price, Moody's seasoned Aaa corporate bond yield, and Moody's seasoned Baa corporate bond to 10-year treasury yield.

The proposed model is established and implemented to forecast prices of the three selected stocks one day ahead. To minimize the impact of uncertainty caused by missing data, we selected three representative stocks with relatively fewer missing values. Missing data were handled using linear interpolation to ensure consistency in the data dimensions across the three stocks. The datasets were then merged into a single file and then divided into training and test sets using a 75% and 25% split. The detailed procedures of establishing the model will not be demonstrated in this case study, since they are already explained in the previous one. Here, only the initial settings are listed as follows. For predictions one day into the future, the model time delay is set at $D = 1$, while the maximum time lags for inputs are established at $d_x = d_y = 10$. Subsequently, historical sequences encompassing all variables are generated and utilized as input features. Following this, the shared feature selection layer identifies 8 key shared features, as listed in Table 5.

In this case, we use a straightforward neural network structure with just one hidden layer to handle all the chosen features. This decision is driven by our primary goal in this study, which is to explain the factors influencing the stock market. Our predictions consistently achieve an accuracy rate of over 90%, as indicated in Table 6. Therefore, there is no pressing need to further boost the model performance. Although our focus is not improving prediction accuracy, it is clear that the shared features and unified structure play a significant role in achieving these accurate predictions. Notably, when it comes to the Village Super Market Inc dataset, individual models perform poorly, indicating high uncertainty in that specific dataset. In contrast, the proposed model consistently delivers strong predictions, suggesting that insights from other stock data can enhance local predictions. This highlights the successful identification of correlations through our proposed approach.

The results presented in Table 5 show that historical stock prices play a crucial role in determining future stock prices. This finding is consistent with the finance literature that past realized stock returns contain valuable pricing information and predict future stock returns [37,38]. In addition, it is noteworthy that previous exchange rates also exert a notably positive influence on future stock price. This finding is in line with the research in the finance literature, emphasizing that the shifts in real exchange rate impact a company's export competitiveness, subsequently affecting profitability. Consequently, this would impact the firm's values and stock prices, especially for companies that engage in international operations [39,40]. More importantly, the interpretability of the model offers fresh insights to the existing literature by underscoring the significance of past stock price and exchange rate for a given day. Specifically, as shown in Table 5, stock prices over the preceding ten trading days, particularly those lagging by one and two days ("Stock Price(t−1)" and "Stock Price (t−2)"), have a stronger predictive power of future stock price. In terms of the past exchange rate, it is worth highlighting that only the US to UK exchange rate with a 4-day lag has a significant impact on stock price for the following day. Taken together, the close correspondence between the model's key features and established theoretical and empirical evidence provides domain-level validation of the model's interpretability, indicating that the shared features capture economic meanings.

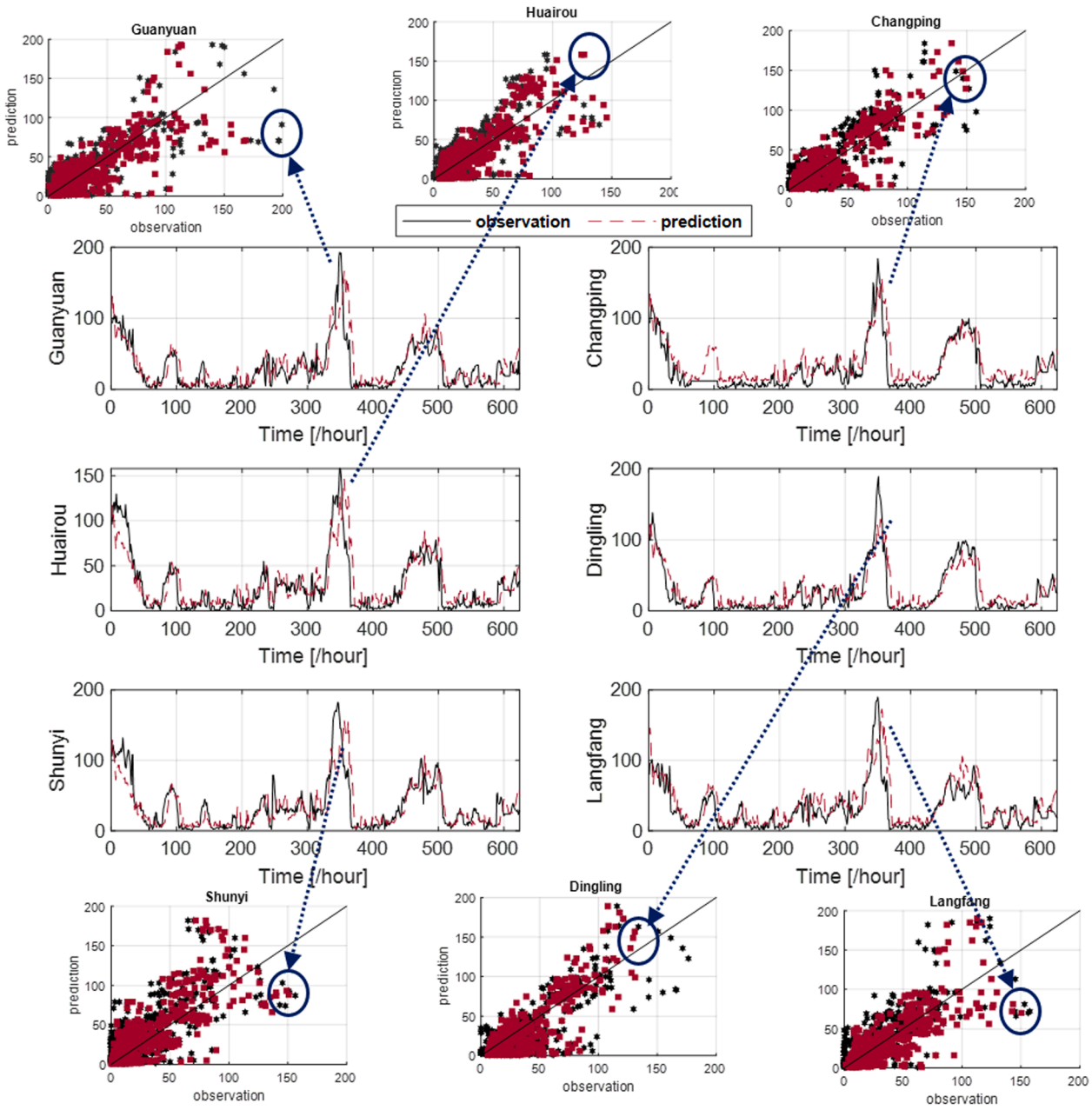

Fig. 7. Comparisons of predictions and observations. The dense plots illustrate the overall predictions, while the scatter plots highlight the predictions for certain severe peak values.

Table 5
Shared features and AER values. Time unit: days.

Index	Feature	AER
1	Stock Price(t−1)	20.5967
2	Stock Price(t−2)	18.7437
3	Stock Price(t−7)	18.2200
4	US to UK exchange rate(t−4)	18.0178
5	Stock Price(t−9)	17.7868
6	China to US exchange rate(t−7) × Stock Price(t−6)	17.6080
7	Stock Price(t−3)	17.1535
8	Stock Price(t−4)	17.0388

Table 6
Prediction performances of the CSNN model and individual models across three representative stock prices. The CSNN model uses the shared features listed in Table 5 to predict all three stock prices, whereas the individual models select separate features for each stock price.

Criterion	Model	Kroger Co	Weis Markets Inc	Village Super Market Inc	Average
Root Mean Square Error	CSNN	0.7147	1.0665	0.8664	0.8825
	Individual	0.7489	1.0721	2.4411	1.4207
Correlation Coefficient	CSNN	0.9970	0.9968	0.9384	0.9774
	Individual	0.9967	0.9968	0.4825	0.8253
Prediction Efficiency	CSNN	0.9940	0.9936	0.8767	0.9548
	Individual	0.9934	0.9936	0.0000	0.6623

5. Conclusion

This article proposed a novel model for shared feature identification and predictive modelling with multiple datasets. The experiments confirm that the proposed CSNN model outperforms traditional predictive models. Its superior performance is largely due to its use of shared features and structure, which enhances predictive accuracy. Comparisons with other models support this improvement. The model's effectiveness is demonstrated through three case studies: one simulation study, one on air pollution, highlighting key factors affecting PM2.5 levels, and another on stock market data, showcasing its versatility in finance. These results suggest that the CSNN model can reduce the need for extensive data collection and improve model accuracy. Future work will explore applying the model to different neural network architectures and other application areas.

Future work will also examine the sensitivity of the shared feature selection mechanism to the choice of missing data imputation strategy. While linear interpolation was employed in this study for its simplicity, more sophisticated approaches such as forward-fill, spline interpolation, or model-based methods may reconstruct feature values differently, potentially influencing the resulting.

CRediT authorship contribution statement

Mao Zhang: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Baihua Li:** Writing – review & editing, Supervision, Resources, Project administration, Funding acquisition, Data curation, Conceptualization. **Qinggang Meng:** Writing – review & editing, Supervision, Resources, Project administration, Funding acquisition, Data curation, Conceptualization. **Yuanlin Gu:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization.

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.

Acknowledgement

This work was partially supported by the British Academy under grant reference SRG24\241444, and the University of Stirling. The authors also gratefully acknowledge the Newton Fund for supporting the early stages of this research under grant reference 104314.

Data availability

Data will be made available on request.

References

[1] L.V. Jospin, H. Laga, F. Boussaid, W. Buntine, M. Bennamoun, Hands-on Bayesian neural networks: a tutorial for deep learning users, IEEE Comput. Intell. Mag. 17 (2) (May 2022) 29–48, https://doi.org/10.1109/MCI.2022.3155327.
[2] S.M. Cabaneros, J.K. Calautit, B.R. Hughes, A review of artificial neural network models for ambient air pollution prediction, Environ. Model. & Softw. 119 (Sep. 2019) 285–304, https://doi.org/10.1016/j.envsoft.2019.06.014.
[3] Z. Li, F. Liu, W. Yang, S. Peng, J. Zhou, A survey of convolutional neural networks: analysis, applications, and prospects, IEEE Trans. Neural Netw. Learn. Syst. 33 (12) (2022) 6999–7019, https://doi.org/10.1109/TNNLS.2021.3084827.
[4] P. Pérez, A. Trier, J. Reyes, Prediction of PM2.5 concentrations several hours in advance using neural networks in Santiago, Chile, Atmos. Environ. 34 (8) (2000) 1189–1196, https://doi.org/10.1016/S1352-2310(99)00316-7.
[5] W.Z. Lu, H.Y. Fan, S.M. Lo, Application of evolutionary neural network method in predicting pollutant levels in downtown area of Hong Kong, Neurocomputing 51 (2003) 387–400, https://doi.org/10.1016/S0925-2312(02)00623-9.
[6] J.B. Ordieres, E.P. Vergara, R.S. Capuz, R.E. Salazar, Neural network prediction model for fine particulate matter (PM2.5) on the US–Mexico border in El Paso (Texas) and Ciudad Juárez (Chihuahua), Environ. Model. & Softw. 20 (5) (2005) 547–559, https://doi.org/10.1016/j.envsoft.2004.03.010.
[7] P.V. de Campos Souza, Fuzzy neural networks and neuro-fuzzy networks: a review of the main techniques and applications used in the literature, Appl. Soft Comput. 92 (Jul. 2020) 106275, https://doi.org/10.1016/j.asoc.2020.106275.
[8] Y. Gu, Y. Yang, J.P.A. Dewald, F.C.T. Van Der Helm, A.C. Schouten, H.L. Wei, Nonlinear modeling of cortical responses to mechanical wrist perturbations using the NARMAX method, IEEE Trans. Biomed. Eng. 68 (3) (Mar. 2021) 948–958, https://doi.org/10.1109/TBME.2020.3013545.
[9] S.A. Billings, 2013, Nonlinear Syst. Identif. NARMAX Methods Time Freq. Spatio-Tempo Domains10.1002/9781118535561.
[10] H.L. Wei, S.A. Billings, Model structure selection using an integrated forward orthogonal search algorithm assisted by square correlation and mutual information, Int. J. Model. Identif. Control. 3 (4) (2008) 341–356, https://doi.org/10.1504/IJMIC.2008.020543.
[11] S. Billings, H.L. Wei, NARMAX model as a sparse, interpretable and transparent machine learning approach for big medical and healthcare data analysis, in Proc. 21st IEEE Int. Conf. High Performance Computing and Communications (HPCC, 2019, pp. 2743–2750, https://doi.org/10.1109/HPCC/SMARTCITY/DSS.2019.00385.
[12] A. Alimissis, K. Philippopoulos, C.G. Tzanis, D. Deligiorgi, Spatial estimation of urban air pollution with the use of artificial neural network models, Atmos. Environ. 191 (Oct. 2018) 205–213, https://doi.org/10.1016/j.atmosenv.2018.07.058.
[13] Y. Gu, B. Li, Q. Meng, Hybrid interpretable predictive machine learning model for air pollution prediction, Neurocomputing 468 (2022) 123–136, https://doi.org/10.1016/j.neucom.2021.09.051.
[14] G. Sambasivam, G.D. Opiyo, A predictive machine learning application in agriculture: cassava disease detection and classification with imbalanced dataset using convolutional neural networks, Egypt. Inform. J. 22 (1) (2021) 27–34, https://doi.org/10.1016/j.eij.2020.02.007.
[15] E.M. Kenny, C. Ford, M. Quinn, M.T. Keane, Explaining black-box classifiers using post-hoc explanations-by-example: The effect of explanations and error-rates in XAI user studies, Artif. Intell. 294 (May 2021) 103459, https://doi.org/10.1016/j.artint.2021.103459.
[16] C. Molnar, G. Casalicchio, B. Bischl, Interpretable machine learning – A brief history, state-of-the-art and challenges, Commun. Comput. Inf. Sci. 1323 (2020) 417–431, https://doi.org/10.1007/978-3-030-65965-3_28.
[17] Y. Gu, B. Li, Q. Meng, P. Shang, Improved Cloud-NARX estimation algorithm for uncertainty analysis of air pollution prediction, Proc. 11th Int. Conf. Pattern Recognit. Syst. (2021).
[18] I.U. Ekanayake, D.P.P. Meddage, U. Rathnayake, A novel approach to explain the black-box nature of machine learning in compressive strength predictions of concrete using Shapley additive explanations (SHAP), Case Stud. Constr. Mater. 16 (Jun. 2022) e01059, https://doi.org/10.1016/j.cscm.2022.e01059.
[19] Y. Gu, H.L. Wei, A robust model structure selection method for small sample size and multiple datasets problems, Inf. Sci. 451–452 (Jul. 2018) 195–209, https://doi.org/10.1016/j.ins.2018.04.007.
[20] Y. Gu, B. Li, Q. Meng, Multi-task deep neural network for air pollution prediction and spatial effect analysis. Proc. 8th Int. Conf. Cloud Computing and Big Data Analytics, IEEE, Jun. 2023, https://doi.org/10.1109/ICCCBDA56900.2023.10154688.
[21] R.C. Gosling, et al., Incorporating clinical parameters to improve the accuracy of angiography-derived computed fractional flow reserve, Eur. Heart J. Digit. Health 3 (3) (Oct. 2022) 481–488, https://doi.org/10.1093/ehjdh/ztac045.
[22] Z. Cheng, L. Li, J. Liu, Identifying the spatial effects and driving factors of urban PM2.5 pollution in China, Ecol. Indic. 82 (Nov. 2017) 61–75, https://doi.org/10.1016/j.ecolind.2017.06.043.
[23] D. Wang, K. Liu, X. Zhang, H. Wang, Spatiotemporal multitask learning for 3-D dynamic field modeling, IEEE Trans. Autom. Sci. Eng. 17 (2) (Apr. 2020), https://doi.org/10.1109/TASE.2019.2941736.
[24] K. Zhang, L. Zheng, Z. Liu, N. Jia, A deep learning based multitask model for network-wide traffic speed prediction, Neurocomputing 396 (2020) 438–450, https://doi.org/10.1016/j.neucom.2018.10.097.
[25] H. Kaji, H. Iizuka, M. Sugiyama, ECG-based concentration recognition with multi-task regression, IEEE Trans. Biomed. Eng. 66 (1) (2019) 101–110, https://doi.org/10.1109/TBME.2018.2830366.
[26] K. Zhang, Z. Liu, L. Zheng, Short-term prediction of passenger demand in multi-zone level: temporal convolutional neural network with multi-task learning, IEEE

Trans. Intell. Transp. Syst. 21 (4) (2020) 1480–1490, https://doi.org/10.1109/TITS.2019.2909571.
[27] K. Fu, H. Li, X. Shi, CTF-former: a novel simplified multi-task learning strategy for simultaneous multivariate chaotic time series prediction, Neural Netw. 177 (2024) 106234, https://doi.org/10.1016/j.neunet.2024.106234.
[28] J. Deng, X. Chen, R. Jiang, X. Song, I.W. Tsang, A multi-view multi-task learning framework for multivariate time series forecasting, IEEE Trans. Knowl. Data Eng. 35 (8) (2023) 7665–7680, https://doi.org/10.1109/TKDE.2022.3218803.
[29] S. Wu, D. Peng, Pre-SMATS: a multi-task learning based prediction model for small multi-stage seasonal time series, Expert. Syst. Appl. 198 (2022) 117121, https://doi.org/10.1016/j.eswa.2022.117121.
[30] J. Wei, X. Wu, T. Yang, R. Jiao, Ultra-short-term forecasting of wind power based on multi-task learning and LSTM, Int. J. Electr. Power & Energy Syst. 149 (2023) 109073, https://doi.org/10.1016/j.ijepes.2023.109073.
[31] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, D.J. Inman, 1D convolutional neural networks and applications: a survey, Mech. Syst. Signal. Process. 151 (2021), https://doi.org/10.1016/j.ymssp.2020.107398.
[32] W. Wei, H. Wu, H. Ma, An autoencoder and LSTM-based traffic flow prediction method, Sensors 19 (13) (2019), https://doi.org/10.3390/s19132946.
[33] B. Lim, S. Zohren, Time-series forecasting with deep learning: a survey, Philos. Trans. R. Soc. A 379 (2194) (Apr. 2021), https://doi.org/10.1098/rsta.2020.0209.
[34] Y. Zhang, Q. Yang, A survey on multi-task learning, IEEE Trans. Knowl. Data Eng. 34 (12) (2022) 5586–5609, https://doi.org/10.1109/TKDE.2021.3070203.
[35] E. De Angelis, C. Carnevale, G.Di Marcoberardino, E. Turrini, M. Volta, Low emission road transport scenarios: an integrated assessment of energy demand, air quality, GHG emissions, and costs, IEEE Trans. Autom. Sci. Eng. (2021), https://doi.org/10.1109/TASE.2021.3073241.
[36] S&P Global Market, Intelligence, S&P Capital IQ Pro Database, New York, NY, USA, 2026.
[37] M. Grinblatt, T.J. Moskowitz, Predicting stock price movements from past returns: The role of consistency and tax-loss selling, J. Financ. Econ. 71 (3) (2004) 541–579, https://doi.org/10.1016/S0304-405X(03)00176-4.
[38] L.-W. Chen, H.-Y. Yu, W.-K. Wang, Evolution of historical prices in momentum investing, J. Financ. Mark. 37 (2018) 120–135, https://doi.org/10.1016/j.finmar.2017.07.001.
[39] N. Ülkü, E. Demirci, Joint dynamics of foreign exchange and stock markets in emerging Europe, J. Int. Financ. Mark. Inst. Money 22 (1) (2012) 55–86, https://doi.org/10.1016/j.intfin.2011.07.005.
[40] H.T. Wong, Real exchange rate returns and real stock price returns, Int. Rev. Econ. & Financ. 49 (2017) 340–352, https://doi.org/10.1016/j.iref.2017.02.004.