IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 11 January 2024, accepted 29 January 2024, date of publication 13 February 2024, date of current version 20 February 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3365499

RESEARCH ARTICLE

A Predictive Adaptive Learning Method for Multivariable Time Series With Mooney Viscosity Prediction as an Application Case

YINGHAO ZHU, CHENHAO YING, YAN ZHOU, AND JIANGANG LU

State Key Laboratory of Industrial Control Technology, College of Control Science and Engineering, Zhejiang University, Hangzhou 310027, China

Corresponding author: Jiangang Lu (lujg@zju.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 62293504 and Grant 62293500, and in part by the National Key Research and Development Plan of China under Grant 2022YFC2403103.

ABSTRACT Time series prediction involves static and dynamic features. Extraneous input information hampers model performance, and the statistical attributes of time series change over time, affecting distribution. Targeted processing of input data for feature and distribution dynamics is vital. This article introduces AdaDynaTrans, an attention-based optimization model. It dynamically combines features from static covariate extraction and temporal variable evolution modules, learning time relationships at various scales. Using self-designed dynamic residual and feature selection units, it suppresses irrelevant information. AdaDynaTrans tackles time covariate shift through temporal segmentation and segmented training mechanisms. Additionally, a relative position-optimized transformer is employed to capture local dependencies within temporal data, thereby achieving exceptional performance within real-world scenarios. Through comprehensive evaluations on both public datasets and industrial scenarios, the considerable efficacy of AdaDynaTrans is demonstrated. Ablation experiments are conducted to analyze the effectiveness of each constituent element, and the interpretability of the model is showcased using a case study involving Mooney viscosity prediction in rubber production. This research contributes substantively to the field by presenting a model that not only achieves high performance but also offers insights into the temporal dynamics of complex industrial processes.

INDEX TERMS Deep learning, time series, attention optimization, Mooney viscosity.

I. INTRODUCTION

Time series data has garnered significant attention across diverse domains, including air pollution prediction [1], time series anomaly detection [2], stock trend prediction [3], renewable energy production [4], and medical monitoring analysis [5]. Classifiable into univariate and multivariate time prediction, the former is amenable to numerous classical prediction methods. Predominantly, classical statistical models such as ARIMA [6], and machine learning models like HMM [7], exponential smoothing [8], Kalman filtering [9], and SVR [10] have been employed. These approaches amalgamate prior knowledge and time series characteristics to yield proficient predictions for linear time prediction. Nevertheless, their applicability faces challenges, particularly in processing large-scale multivariate time series data. The limitations arise from the inability to effectively integrate time-related features and the substantial computational overhead required for the convergence of machine learning algorithms when dealing with extensive datasets. As a consequence, despite their efficacy in certain contexts, these methods encounter constraints that impede their widespread adoption in practical applications.

To enhance predictive performance and surmount challenges inherent in multivariate time series prediction within the big data paradigm, this study employs a recurrent neural

The associate editor coordinating the review of this manuscript and approving it for publication was Claudio Zunino.

network (RNN) [11] to glean temporal correlations. However, RNN encounters issues of gradient vanishing and explosion when processing extended time series due to its incapacity to selectively retain or forget prior hidden states. To ameliorate these concerns, LSTM [12] and GRU [13] models have been developed, prioritizing short-term dependencies through distinct gating mechanisms that modulate information flow between current and previous hidden states. BiLSTM (Bidirectional Long Short-Term Memory Network) [14], an extension of LSTM with reverse computation, enhances correlation across temporal information. Despite these advancements, RNN networks [15] struggle to emphasize key time series information, thereby impacting prediction accuracy. In contrast, transformer [16], diverging from the cyclic network mechanism of RNN, do not rely on prior state inputs.Instead, transformer analyzed complete time series inputs, leveraging the attention mechanism to comprehensively learn temporal relationships, rendering them adept at handling long-term dependencies. Operations such as position encoding, padding mask, and look-ahead mask enhance their capability to capture short-term dependency relationships in time series. Consequently, transformer-based models demonstrate proficiency in simulating the intricate dynamics of multivariate time series. However, their effectiveness is predominantly assessed using standard public datasets, with limited successful applications in real-world scenarios.

The overall process of rubber compounding includes the following stages: feeding, immersion, dispersion, mastication, and discharge. This process involves the mixing of various rubber materials, with the rubber processing primarily encompassing the stages of immersion, dispersion, and mastication. In this study, the rubber processing stage is considered the main focus, and the term ''rubber compounding'' in the following text specifically refers to these three stages. The internal mixer utilizes a motor to drive the rotor for relative rotation, generating forces. During the operational process, the motor experiences fluctuations in parameters such as equipment current, compounding chamber temperature, and equipment power. As compounding time progresses, different types of rubber materials undergo friction between the chamber walls and rotor in the internal mixer, generating heat and causing continuous changes in the temperature and state of the rubber materials. Therefore, it can be understood that there are potential correlations among the process parameters, such as rotor speed, compounding power, compounding time, and compounding temperature. If the underlying patterns among these features can be identified, it enables mastery of control techniques for process parameters, thereby positively impacting the final rubber quality.

Mooney viscosity [17] serves as an intuitive feedback indicator for assessing the quality of rubber tempering, pivotal in determining whether the physical properties of rubber align with prescribed standards. Each stage of rubber tempering necessitates distinct processing operations, resulting in notable disparities in dynamic time series variables (e.g., capacity, current) and influenced by static characteristics of pre-tempering processing (e.g., raw material ratio, equipment number) [18]. Consequently, the prediction of Mooney viscosity can be construed as a multivariate time series prediction challenge. Presently, the primary approach for predicting Mooney viscosity [19] relies on sensors and manual monitoring of temperature, processing time, and other characteristic changes in the machinery. However, in the actual process workflow, there exists a delay between completing rubber processing and predicting Mooney viscosity. Immediate determination of Mooney viscosity following rubber refining could significantly impact process flow optimization. In real industrial scenarios, diverse and intricate time series challenges prevail, with predicting Mooney viscosity during rubber tempering representing just one facet of these challenges. Addressing the intricacies of multivariate time series prediction involves targeted processing of different feature categories and data distributions. This approach enables a comprehensive understanding of the relationships between inputs strongly linked to the prediction target, facilitating accurate predictions in complex industrial settings.

This study aims to substantiate the predictive efficacy of AdaDynaTrans through a meticulous evaluation leveraging a publicly accessible dataset. The specific context of Mooney viscosity prediction within authentic industrial scenarios serves as a focused research case for assessment. Notably, the findings underscore AdaDynaTrans's superiority over prevailing prediction methods. Subsequent ablation experiments were conducted to ascertain the affirmative influence of each model component on the ultimate prediction accuracy. Our contributions are distinctly outlined as follows:

1) Feature Processing Enhancement: A novel approach is introduced from the feature processing dimension, involving the design of Dynamic Residual Units (DRU) and Feature Focused Units (FFU). These units traverse the entire gating mechanism and feature selection process, effectively minimizing weakly correlated segments of the multivariate input with the predicted target.
2) Data Distribution Consideration: The study incorporates temporal segmentation and segmented training (TSST) strategies to mitigate the impact of time series covariate shifts on prediction outcomes. Mooney viscosity prediction serves as a practical experiment, offering a specific use case for real-world industrial scenarios.
3) Transformer-Based Time Series Prediction Model: A transformative model is proposed, leveraging a fully connected directed graph to refine the attention mechanism. This enhancement augments the model's comprehension of relative position information within the input time series. The preprocessing of diverse feature types further contributes to the overall improvement in predictive performance.

The ensuing sections of this article are structured as follows: Section II delves into pertinent research pertaining to time series prediction, offering a comprehensive overview of the existing landscape. Section III presents an intricate elucidation of the diverse components and underlying principles integral to the proposed model. Subsequently, Section IV systematically conducts ablation experiments utilizing the Mooney prediction dataset, thereby substantiating the efficacy of each model component. Finally, Section V encapsulates the findings and draws conclusive insights, providing a nuanced culmination to this scholarly exploration.

II. RELATED WORK

A. TIME SERIES PREDICTION

Time series prediction holds widespread applicability across industries, medicine, commerce, and diverse domains. Conventional modeling approaches rely on statistical models grounded in professional knowledge, whereas machine learning presents a data-driven paradigm for understanding temporal dynamics [20]. The integration of deep learning, with tailored architectural assumptions reflecting nuanced dataset differences, enables the extraction of intricate data representations [21]. In the realm of time series research, recurrent neural networks, exemplified by GRU and LSTM, stand out for their capacity to autonomously extract high-quality features and adeptly manage long-term dependencies within temporal data. A prevailing trend involves leveraging diverse network models in deep learning to amalgamate distinct advantages and capture potential relationships in time series. For instance, LSTNet [22] integrates convolutional neural networks and recurrent neural networks to discern short-term local dependency patterns between variables, addressing the rule insensitivity issue of neural networks through traditional autoregressive models.

Additionally, a focus on data denoising and distribution perspectives in modeling identifies inputs with the strongest correlation with the predicted target. AdaRNN [23], pioneering the Time Series Covariate Problem (TCS), resolves TCS from the vantage point of data distribution using adaptive RNN. Its performance surpasses robust time series baseline models like STRIPE [24] in public datasets. Google's TFT [25], grounded in the attention mechanism, learns time relationships across different scales from the standpoint of data features. It seamlessly integrates multi-horizon prediction and interpretable insights into temporal dynamics, showcasing notable performance enhancements in real-world datasets such as finance and traffic prediction. This landscape illustrates the evolving and multifaceted approaches within the contemporary milieu of time series prediction research.

B. RUBBER MIXING PROCESS

Rubber mixing, as described in empirical studies such as [20], encompasses a process wherein diverse ingredients are dispersed and mixed within raw rubber via an internal mixer. The primary objective of this procedure is to achieve thorough mixing and dispersion of various raw materials, refine the uniform distribution of ingredients, ultimately resulting in a compound undergoing both physical and chemical changes [26]. The sequential stages of this process involve feeding, soaking, dispersing, kneading, and discharging, with the central focus of rubber processing concentrated on the intervening dispersing, kneading, and discharging stages [27].

The internal mixer operates by generating relative rotational forces through a pair of rotors driven by an electric motor. The motor's operational dynamics introduce fluctuations in parameters such as current and energy. Over time, interactions involving friction and heat generation manifest among various rubber materials, between these materials and the internal mixer's chamber wall, and among the rotors. Consequently, there is a continual evolution in the temperature and state of the rubber material. Implicitly, potential connections exist between rotor speed, rubber mixing power, mixing time, mixing temperature, and other characteristics within the parameters of the rubber mixing process. The excavation of potential relationships between these characteristics holds the promise of advancing control technology for process parameters, thereby exerting a positive influence on the ultimate quality of the rubber material.

C. PROBLEM DEFINITION

Consider a Mooney viscosity dataset comprising I samples, each derived from a comprehensive production record during the rubber mixing process. Each sample i encompasses a static variable s_i, a time series feature x_i spanning t_N instances ($i = 1, \ldots, t_N$), and a target variable $\widehat{y}$. The prediction of mooney viscosity, denoted as $\hat{y}_i = f(si, x_i)$, is formulated with $\hat{y}_i$ representing the predicted Mooney viscosity value and $f(\cdot)$ symbolizing the proposed AdaDynaTrans model introduced in this study. This Mooney viscosity prediction task is inherently supervised, wherein each input sample x_i comprises multiple time series elements of length t_N, and the static covariate s_i encapsulates multiple scalar features.

III. MODEL ARCHITECTURE

Time series prediction problems inherently involve diverse types of input features. Conventional approaches often overlook this diversity and directly engage in feature processing. However, this indiscriminate treatment frequently yields prediction outcomes that significantly deviate from actual scenarios. The intrinsic noise within time series data further complicates subsequent modeling and prediction tasks. In our proposed model, AdaDynaTrans, we adopt a discerning approach by classifying input features, thereby facilitating the extraction of features highly correlated with the prediction task. This classification-based processing serves to mitigate the impact of noise inherent in time series data. On the level of data distribution, we introduce a temporal segmentation and segmented training mechanism, culminating in the development of an Ada Position-Enhanced Former. This

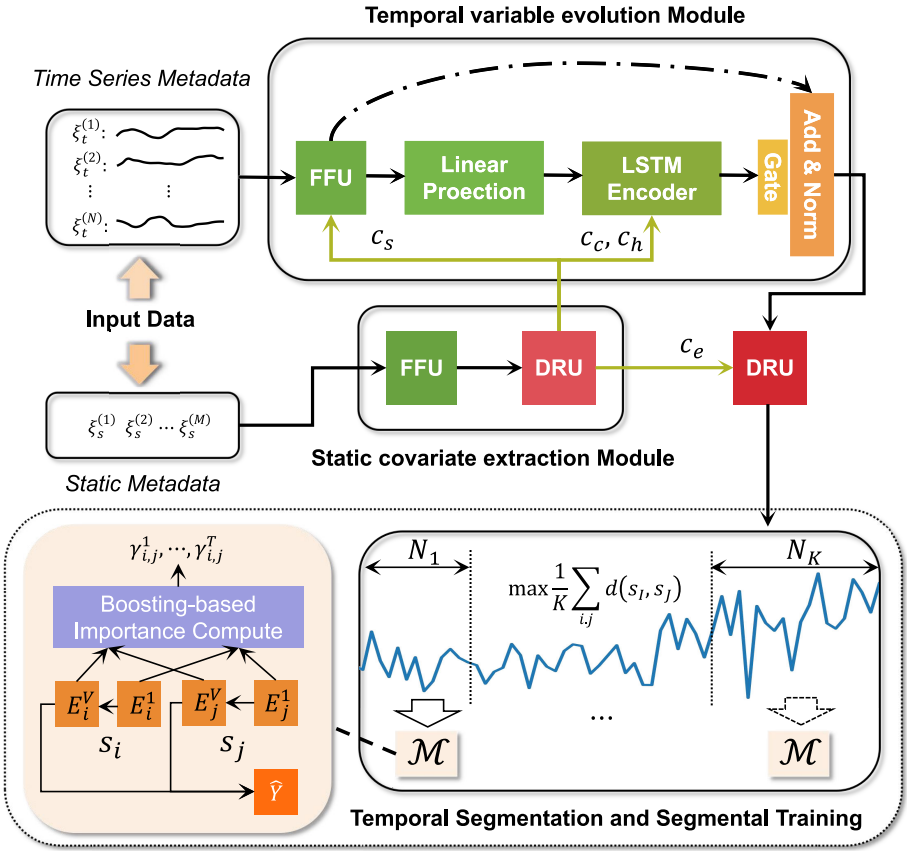


FIGURE 1. AdaDynaTrans structure diagram.

entails determining optimal time segmentation by maximizing similarity between different periods. Subsequently, the model is employed to learn the distribution within each period under worst-case conditions. The acquired knowledge is then leveraged for prediction using the learned model and new data. This nuanced approach aims to enhance the accuracy and robustness of time series predictions within the proposed AdaDynaTrans model.

The structural diagram of AdaDynaTrans is shown in Figure 1, and the main components that make up AdaDynaTrans are as follows:

Dynamic Residual Unit (DRU): It is a network component based on the gating mechanism, with ELU serving as the activation function between two fully connected layers. Subsequently, GLU (Gated Linear Unit) is introduced, and the original input is added to the output of GLU through skip connections. The resulting sum undergoes layer normalization before being output. This architecture can flexibly adapt to various datasets and scenarios, and achieve the function of automatically adjusting network depth and complexity by skipping unused components.

Feature Focused Unit (FFU): This component is constructed based on DRU (Dynamic Recurrent Unit). It comprises (input feature number + 1) DRUs, each of which transforms the input features accordingly. The combination of all input features is then unfolded into a long vector. After adding this vector to external environmental variables, it is input into an independent DRU. Following a softmax operation, the result is transformed into a weighted vector, where each feature corresponds to a specific weight. The output is a weighted vector representing each feature and its corresponding weight. Select the most noteworthy variable features of the network at each time step to achieve denoising effect.

Static Covariate Extraction Module (SCEM): Built upon DRU (Dynamic Recurrent Unit) and FFU (Feedforward Unit), this architecture consists of one FFU and one DRU. By invoking the DRU component four times, four distinct context vectors are generated. Integrate static features into the network and achieve the effect of adjusting temporal dynamics by encoding context vectors.

Temporal Variable Evolution Module (TVEM): This component comprises one FFU, followed by a linear mapping layer and a 4-layer LSTM Encoder. Simultaneously, the output of the FFU is weight-normalized through skip connections with the LSTM's output, ultimately serving as the output for this module. Integrating time series features into the network, accepting contextual variables provided by SCEM at different stages to add to the network, dynamically enriching the semantics of time series features.

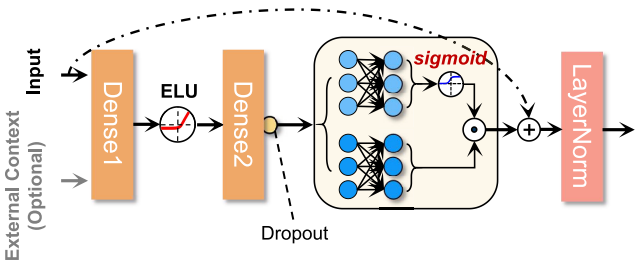


FIGURE 2. Dynamic residual unit network diagram.

Temporal Segmentation and Segmental Training (TSST): According to the principle of maximum entropy, the temporal distribution is divided into multiple segments with the least similarity to obtain diversity in the time series data. By matching the contribution weights of the Transformer's Encoder in different segments, common knowledge in different distributions is learned more comprehensively, enhancing the generalization of the model.

Position Enhanced Transformer (PET): This component replaces the original multi-head attention mechanism in the Transformer with an optimized multi-head attention mechanism. In order to capture the relative position information of each element in the input time series data, the relative position information of the element is added to the multi head attention of the transformer in a fully connected directed graph.

A. DYNAMIC RESIDUAL UNIT

The precise relationship between external inputs and targets is often elusive, posing challenges in discerning relevant variables and determining the optimal degree of non-linear treatment for these variables. This ambiguity is particularly pronounced in scenarios with limited datasets or high levels of noise, where simpler linear models may outperform more complex counterparts. As illustrated in Figure 2, to endow our model with the capability to selectively apply nonlinear processing when deemed necessary, we have devised a Dynamic Residual Unit (DRU) as the foundational module of AdaDynaTrans. The DRU takes a primary input a and an optional context vector c, producing outputs that enhance the model's adaptability to varying degrees of non-linearity. This architectural choice ensures a flexible and adaptive approach to processing external inputs, contributing to the model's effectiveness across diverse scenarios.

$$\mathrm{DRU}_{\mathrm{out}} = \mathrm{LN}\big(a + \mathrm{GLU}(hid_2)\big) \tag{1}$$

$$hid_2 = W_3 hid_1 + b_2 \tag{2}$$

$$hid_1 = \begin{cases} \mathrm{ELU}(W_1 a + W_2 c + b_1), & c \neq 0 \\ \mathrm{ELU}(W_1 a + b_1), & c = 0 \end{cases} \tag{3}$$

Among them, ELU is the exponential linear unit activation function, hid_i represents the intermediate layer, and LN is the layer normalization. When $W_a + W_2 c + b_1 \gg 0$, ELU will act as an identity function, and when $W_a + W_2 c + b_1 \ll 0$, ELU will produce a constant output. In order to flexibly adjust the model structure and suppress irrelevant components for a given dataset, we adopt a component gated network based on gated linear units (GLU). In this way, we can selectively adapt and simplify the model based on the characteristics of different datasets. The form of GLU can be represented as follows:

$$\mathrm{GLU} = \sigma(W_4 hid_2 + b_3) \odot (W_5 hid_2 + b_4) \tag{4}$$

where $\sigma(.)$ is the activation function, $W(.)$ and $b(.)$ represent the weight and bias of the hidden layer, respectively. Through GLU regulation, AdaDynaTrans can control the degree of influence of DRU on the original input a, and even skip this layer completely if necessary, as the output of GLU may be close to 0, effectively suppressing the influence of nonlinearity. As shown in Formula (3), for the case where there is no context vector, DRU will be set to 0.

B. FEATURE FOCUSED UNIT

Despite the potential presence of numerous variables, their intercorrelation and individual contributions to the predicted target remain ambiguous. AdaDynaTrans endeavors to systematically identify static covariate features and time series features through the FFU. This stepwise feature selection mechanism not only enables the model to pinpoint the most relevant features for prediction but also empowers AdaDynaTrans to eliminate extraneous noise inputs. Such noise inputs typically exert a pronounced negative impact on prediction accuracy. Given that time series datasets commonly exhibit a limited number of features, the FFU's selective feature inclusion enhances the predictive performance of the model by leveraging the most significant features pertinent to the prediction task. This strategic feature selection process contributes to the precision and efficacy of AdaDynaTrans in handling complex time series prediction challenges.

As shown in Figure 3, entity embedding is used as the feature representation for category features, and linear transformation is used for continuous features. Each input variable is transformed into a d_{model} dimension vector to adapt to the jump connected dimensions in subsequent layers, d_{model} represents the dimension of the word vector after embedding. It should be further explained that, as shown by the FFU in TVEM in Figure 1, this component accepts c_s output from SCEM as the external environment variable. For the FFU in TVEM, the DRU on the left side of Figure 3 does not use context variables and directly inputs the original features to obtain the transformed features. For the DRU on the right, all features are combined and added to the context vector cs through softmax to obtain the corresponding weights for each feature. In the FFU components of TVEM and SCEM in Figure 1, We use FFU with different colors to represent static covariate features and time series features separately. In Figure 3, both static covariate features and time features are mapped to the same dimension through DRU and subjected to the same weighting operation.

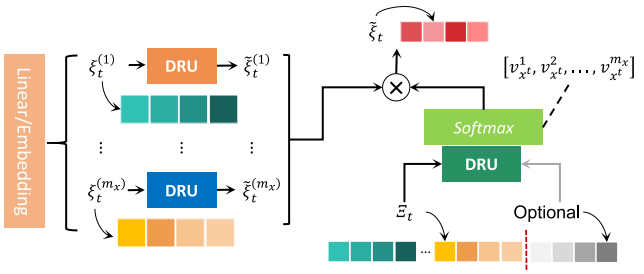


FIGURE 3. Network diagram of feature focused unit.

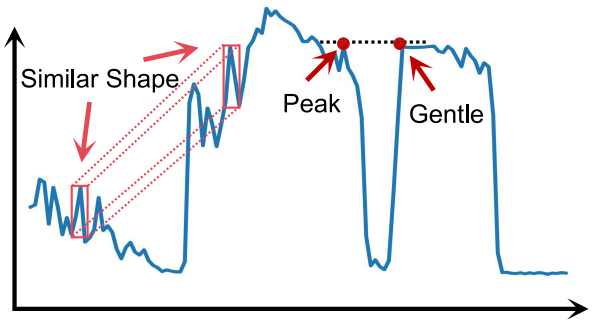


FIGURE 4. Comparison results of applying LSTM Encoder to timing input.

Let $\xi_t^{(1)^T} \in \mathbb{R}^{d_{model}}$ denote the input for the j-th feature at time t, and $\Xi_t = \left[\xi_t^{(1)^T}, \cdots, \xi_t^{(m_x)^T}\right]^T$ represent the flattened vector encompassing all past inputs at time t. Concurrently incorporating Dynamic Residual Unit (DRU) and generating context vectors for feature selection weights, this process is succinctly represented as Formula (5):

$$v_{x_t} = softmax(\mathrm{DRU}_{v_x}(\Xi_t, c_s)) \tag{5}$$

where v_{x_t} is the vector of feature selection weights, and at each time step, by inputting $\xi_t^{(1)^T}$ into the corresponding DRU, an additional nonlinear processing layer is used, as shown in Formula (6)

$$\tilde{\xi}_t^j = \mathrm{DRU}_{\tilde{\xi}_j}\left(\xi_t^{(j)}\right) \tag{6}$$

among them, $\tilde{\xi}_t^j$ is the feature vector processed by variable j. By weighting the processed features with feature selection weights and combining them, Formula (7) is obtained:

$$\widetilde{\xi}_t = \Sigma_{j=1}^{m_x} v_{x_t}^{(j)} \widetilde{\xi}_t^{(j)} \tag{7}$$

where $v_{x_t}^{(j)}$ is the j-th element of vector v_x.

The specific process of weighting each feature in Formula (5) can be explained as depicted in Figure 3. At a given time t, each input feature undergoes a DRU to yield a new vector (depicted by distinct color sequences). After combining these new vectors and flattening them, the vector from c_s is added. Subsequently, this combined vector is transformed into corresponding weights for each vector through DRU and softmax. These weights, along with the vectors, are then subject to weighting and summation, resulting in an entirely new feature vector. This vector integrates the selected information from all input feature vectors, constituting a highly abstract set of input features.

C. STATIC COVARIATE EXTRACTION MODUL

In the processing of other time series prediction models, time series may lose internal local dependencies and relative position information. In order to fully utilize the potential information contained in various features, AdaDynaTrans uses a separate DRU encoder to generate four different context vectors, namely c_s, c_c, c_h, and c_e.

These contextual variables will be integrated into various positions of the static time series feature processing module, as follows:

1) FFU has feature selection capabilities, and by introducing environmental variables, this component can selectively filter all features with the purpose of focusing on content relevant to the final prediction. The variable c_s is employed to assist this component in feature selection, with the specific manifestation that more important features have a larger proportion in terms of feature weights.
2) In practical applications, LSTM requires the specification of initial states for subsequent training and weight updates. The context environment vectors c_c and c_h serve to provide corresponding initial states for LSTM during training. Furthermore, these initial states retain static covariate hidden information from SCEM, facilitating the model in capturing dependencies between temporal elements more effectively.
3) Due to some degree of information loss during the fusion and transmission of various feature information, many detailed pieces of information cannot be adequately supplemented. To better utilize the information within the features, combining c_e with the output of TVEM is advantageous for compensating for missing detailed information, further enriching temporal features.

D. TEMPORAL SEGMENTATION AND SEGMENTAL TRAINING

1) TEMPORAL SEGMENTATION

The data in the input time series processing module is mapped into a vector of the specified dimension after passing through a linear projection layer. The infiltration, kneading, and other operations during the rubber mixing process change over time, especially during the alternating stages of each stage, where the fluctuations in data are more pronounced, as shown in Figure 4. If the data passing through this module is directly input into the subsequent Transformer, the attention score in the attention mechanism is only determined by the Q and K after input feature mapping, and cannot fully utilize the dependency relationships in the previous temporal data. If the local context is unknown and the attention mechanism observes changes in key points, it is difficult to distinguish

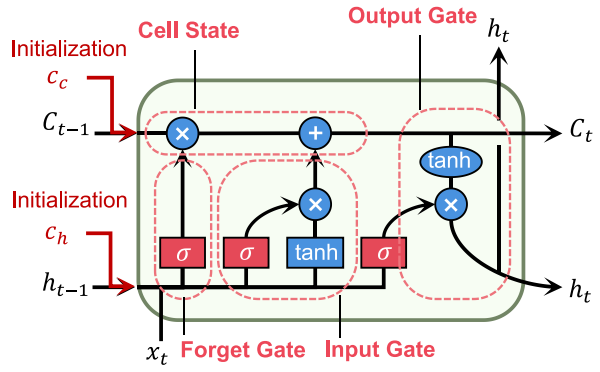


FIGURE 5. LSTM encoder initialization structure diagram.

whether this is a change in points or a component of patterns, thus making accurate prediction difficult.

To alleviate the problem of insufficient capture of local dependency relationships in input data, an LSTM Encoder with four interaction layers is used to perform feature crossover on the linearly mapped dimensions. By using LSTM, the attention mechanism in the Transformer can more easily capture changes in key points in the local environment.

For a time series input, c_c and c_h are context vectors from the static covariate processing module, which are used to initialize the hidden state of the LSTM Encoder. The initialization operations of the LSTM Encoder on the corresponding time element x at time t, with $h_{t-1} = c_h$ and $C_{t-1} = c_c$, are as follows:

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \tag{8}$$

Formula (8) is the operation of the forgetting gate, where w_f is the weight matrix, b_f is the bias term, and σ is the sigmoid function;

$$\widetilde{C}_t = tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{9}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{10}$$

Formula (9) and Formula (10) are input gate operations, where $\widetilde{C}_t$ and i_t represent candidate values and input gate values, W_c and W_i are weight matrices, b_c and b_i are bias terms, h_{t-1} is the hidden state at the previous time, x_t is the current input after linear mapping, and tanh is a hyperbolic tangent function;

$$C_t = C_{t-1} * f_t + i_t * \widetilde{C}_t \tag{11}$$

Formula (11) represents the operation of memory cell update, where f_t is the output of the forgetting gate and C_{t-1} is the cell state at the previous moment;

$$o_t = \sigma(W_0 \cdot [h_{t-1}, x_t] + b_0) \tag{12}$$

$$h_t = o_t * tanh(C_t) \tag{13}$$

Formula (12) and Formula (13) represent the operations of the output gate and hidden state, respectively, where W_0 is the weight matrix, b_0 is the bias term, and h_{t-1} is the hidden state at the previous time.

The internal state of LSTM encoder initialization is shown in Figure 5.

2) SEGMENTAL TRAINING

According to the maximum entropy, it is reasonable to diversify the distribution of each cycle as much as possible in order to maximize the entropy of the total distribution without prior assumptions about the segmentation of time series data. Due to the lack of prior information on the test data, it is not possible to determine the distribution of the test data. Therefore, it is reasonable to train the model under the worst-case segmentation of time series data. If the model can work from the worst-case scenario, it will have better generalization ability for test data. Specifically, given a time series data S with N labeled periods, it is assumed that it can be divided into K time periods, i.e. $S = S_1, \ldots, S_K$, where $S_K = \{x_i, y_i\}_{i=n_K+1}^{n_{K+1}}$, $N_1 = 0$, $N_{K+1} = N$. To achieve the goal of segmenting time series by solving an optimization problem. The objective of this optimization problem can be expressed as:

$$0 < \overset{\max}{K} \leq K_0 \, \overset{\max}{N_1, \ldots, N_K} \frac{1}{K} \sum_{1 \leq i \neq j \leq K} d\left(S_i, S_j\right)$$

$$s.t. \forall i, \Delta_1 < |S_i| < \Delta_2; \sum_{i=1}^{K} |S_i| = N \tag{14}$$

among them, Δ_1 and Δ_2 are predefined parameters to avoid capturing distribution information with very small or large values, K_0 is a hyperparameter to avoid over segmentation of time series data, K represents the k-th cycle to be divided, N_i represents the length of the i-th cycle to be divided, N represents the total length of the time series, and $d\left(S_i, S_j\right)$ represents the distribution distance between the two cycles, which is calculated by cosine similarity.

For a given time series, it is divided into an average of 12 segments, each of which is an indivisible minimum period. Randomly select the value of K from the set 2, 3, 5, 7, 10, 12. After determining K, select each cycle with a length of based on a greedy strategy. Use A and B respectively to represent the starting and ending points of the time series. First, consider K=2 and select 1 splitting point C from 9 candidate points by maximizing the distribution distance d. After determining C, consider K=3 and use the same strategy to select another splitting point D. Similar strategies are applied to different K values, and the optimal K value of the present invention is 4. When K is other values, the final performance of the prediction model will deteriorate.

For the K cycles divided based on the maximum entropy principle, learn common knowledge shared by different time periods by matching the distribution of different time periods. The segmented learning mechanism can better capture the contribution of different distributions to the shared knowledge of model training by adapting and matching the distributions between Encoder units of two Position Enhanced Trans. The segmented learning mechanism loss

used for prediction can be expressed as Formula (15).

$$\mathcal{L}_{pred}(\theta) = \frac{1}{K}\sum_{j=1}^{K}\frac{1}{|S_j|}\sum_{i=1}^{|S_j|}\ell\left(y_i^j\mathcal{M}\left(x_i^j;\varphi\right)\right) \quad (15)$$

among them, (x_i^j, y_i^j) represents the i-th marker segment of period S_j, $\ell(.)$ represents the loss function, φ is a learnable model parameter, and $\mathcal{M}$ is the learning model. Introduce the relative importance of the internal encoding and decoding state V of the R-learning transformer, where all internal output states are weighted by their corresponding normalized $\gamma \in \mathbb{R}^V$. For a given period (S_i, S_j), the loss of segmented learning mechanism can be expressed as Formula (16):

$$\mathcal{L}_{encoder}(S_i, S_j; \varphi) = \Sigma_{t=1}^{V}\gamma_{ij}^t d\left(E_i^t, E_j^t; \varphi\right) \quad (16)$$

where γ_{ij}^t represents the importance distribution between state t and S_i of S_j, and $\mathrm{H} = \{E^t\}_{t=1}^V \in \mathbb{R}^{V\times q}$ represents the V outputs of the transformer encoder with feature dimension q.

By integrating Formula (15) and Formula (16), the ultimate goal of segmented learning is Formula (17):

$$\mathcal{L}(\varphi, \gamma) = \mathcal{L}_{pred}(\varphi) + \mu\frac{2}{K(K-1)}\sum_{i,j}^{i\neq j}\mathcal{L}_{encoder}\left(S_i, S_j; \varphi, \gamma\right) \quad (17)$$

where μ is the weight hyperparameter, in the second term, we calculate the average distribution distance of all pairwise cycles. Formula (17) can be used to perform segmented training mechanism, and γ is learned by boosting based importance evaluation algorithm [23].

E. POSITION ENHANCED TRANSFORME

Transformers play an important role in fields such as natural language processing and computer vision, but it is well known that transformers lack relative position information modeling in multi head attention mechanisms, so there is no significant emphasis on characterizing local dependencies. In order to fully explore the dependencies and relative position perception in time series data, we propose a directed graph fully connected approach to optimize relative position information. Based on this, we optimize the multi head attention mechanism and effectively integrate it into the encoder and decoder. The network structure of PET is shown in Figure 6.

1) IMPROVEMENT OF ATTENTION MECHANISM

The input of the Transformer is the sum of the time series input and the corresponding position encoding. When calculating the attention score of any two elements in the time series, the formula is:

$$\begin{aligned}\mathrm{Attn}_{ij} &= \left(W_q(x_i + PE_i)\right)^T\left(W_k(x_j + PE_j)\right)\\ &= x_i^T W_q^T W_k x_j + x_i^T W_q^T W_k PE_j\\ &\quad + PE_i^T W_q^T W_k x_j + PE_i^T W_q^T W_k PE_j\end{aligned} \quad (18)$$

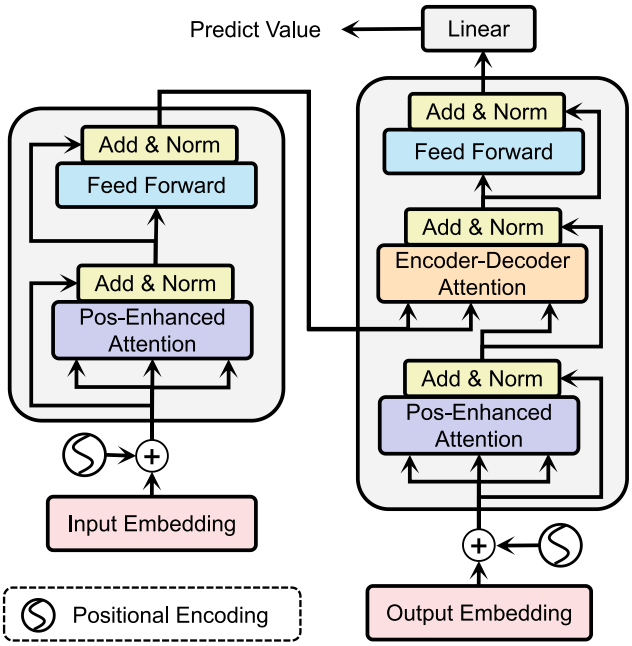


FIGURE 6. PET network structure diagram.

where W_q and W_k are the weight matrices, PE_i and PE_j represent the positional encoding of elements x_i and x_j. Formula (18) consists of four parts, only containing both PE_i and PE_j in the fourth term. Therefore, there may be relative position information of elements x_i and x_j. According to [16], the elements in the time series can be encoded as:

$$\begin{aligned}PE_{(t,2i)} &= \sin\left(\frac{t}{10000^{2i/d_{\text{model}}}}\right)\\ PE_{(t,2i+1)} &= \cos\left(\frac{t}{10000^{2i/d_{\text{model}}}}\right)\end{aligned} \quad (19)$$

where t represents the position of the t-th element in the time series, and i represents the dimension. If μ_i is a constant with a value of $1/10000^{2i/d_{model}}$, k represents the interval between any two elements, so the product of the encoding of different element positions in the time series can be calculated:

$$\begin{aligned}PE_t^T PE_{t+k} &= \sum_{i=0}^{d_{model}/2-1}\left[\sin(\mu_i t)\sin\left(\mu_i(t+k)\right)\right]\\ &\quad + \sum_{i=0}^{d_{model}/2-1}\left[\cos(\mu_i t)\cos\left(\mu_i(t+k)\right)\right]\\ &= \sum_{i=0}^{d_{model}/2-1}\cos\left(\mu_i(t-(t+k))\right)\\ &= \sum_{i=0}^{d_{model}/2-1}\cos(\mu_i k)\end{aligned} \quad (20)$$

the derivation of Formula (20) is based on $\cos(x-y) = \sin(x)\sin(y)+\cos(x)\cos(y)$. According to the above equation, it can be inferred that the product of PE_i and PE_j is

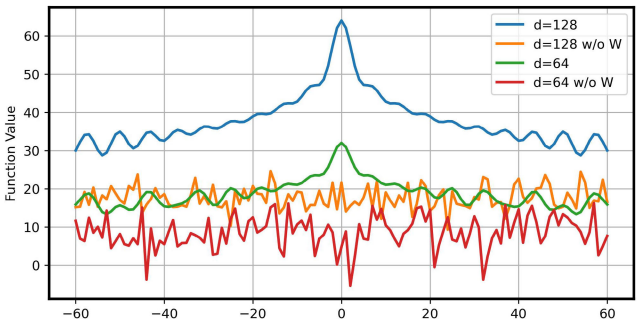


FIGURE 7. Product value curves under different dimensions and weights.

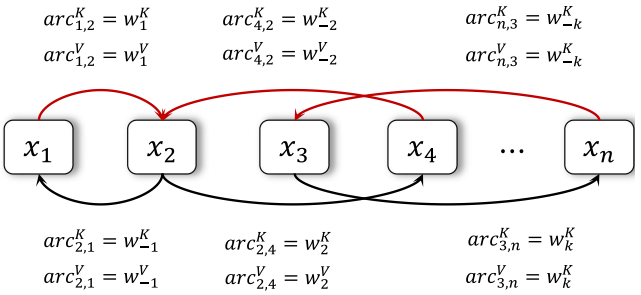


FIGURE 8. Specific example edges of fully connected directed graphs.

only related to the relative positions of the two positions, and this relationship equation contains information about the relative positions between elements. When the fourth term in Formula (18) includes weight parameters W_q and W_k, the relative position information between elements will be masked. To further verify the above conclusion, taking d_{model}=64 and d_{model}=128 as examples, Figure 7 shows the product structure of formulas $PE_t^T PE_{t+k}$ and $PE_i^T W_q^T W_k PE_j$. With the help of images, it is not difficult to find that the result curve of $PE_t^T PE_{t+k}$ has obvious patterns, while the result curve of $PE_i^T W_q^T W_k PE_j$ is disorderly and does not have obvious patterns. Therefore, it was verified that although the transformer added position encoding to the input, it is difficult for the model to capture the relative position information in the time series in practical operations.

In order to add the relative positions of elements in temporal data to the calculation process of the model, we propose the idea of a directed fully connected graph, which treats each element in temporal data as a node and uses directed edges to describe the relative position relationship between elements. Referring to Figure 8, for any two elements x_i and x_j, set two opposite edges arc_{ij}^K and arc_{ij}^V to connect, and each node's two edges represent a trainable weight, thereby adding relative position information to the model during the training process. It is worth noting that in a directed fully connected graph, the reason for $2 \leq k \leq n-5$ is that when the distance between two elements in the time series exceeds a certain value, the impact on the final prediction result is negligible. Therefore, the maximum effective distance between elements is limited to the adjustable parameter k. The relative position information between elements can be expressed by the following formula:

$$\begin{aligned} arc_{ij}^K &= w^K_{clip(j-i,k)} \\ arc_{ij}^V &= w^K_{clip(j-i,k)} \\ clip(x,k) &= \max(-k, \min(k,x)) \end{aligned} \tag{21}$$

among them, $w^K = \left(w^K_{-k}, .., w^K_k\right)$ and $w^K = \left(w^V_{-k}, .., w^V_k\right)$ store the relative position information of each element in the time series data. After the encoding layer inputs through the Position Enhanced layer, a weight matrix of the relative position information between all elements is generated, which is added to the original input and transmitted into the attention mechanism.

2) POSITION ENHANCED MULTIHEAD ATTENTION

The self attention layer in transformer uses multiple attention heads, concatenates the structures of all heads, and then uses linear transformations for transformation. Each attention head consists of a sequence $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n)$ composed of n elements as input, and the calculation results output a new sequence $\mathbf{z} = (\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n)$ of the same length. Each output element can be expressed as Formula (22):

$$z_i = \sum_{j=1}^n \omega_{ij}(x_j W^{\mathcal{V}}) \tag{22}$$

for each weight coefficient ω_{ij}, $\omega_{ij} = e_{ij} / \sum_{\mathbf{k}=1}^{\mathbf{n}} \exp e_{ik}$ can be used to calculate, and the attention score e_{ij} between the i-th and j-th elements can be calculated by Formula (23).

$$e_{ij} = \frac{(x_i W^Q)\left(x_j W^K\right)^T}{\sqrt{d_z}} \tag{23}$$

among them, w_Q, w_K, and w_V are weight matrices, Q represents the query, K represents the correlation vector between the queried information and contextual information, V represents the queried vector, which are used to calculate self attention. d_z represents the dimension of the word vector after embedding, and scaling dot product is used between Q and K to calculate attention scores for efficient calculation.

By utilizing the edge weights of a directed graph to enrich the relative position information between elements, the attention mechanism in the transformer can be optimized. The optimized multi head attention mechanism is called position enhanced multi head attention. Add relative position information to the original attention mechanism. Specifically, the directed edge weights arc_{ij}^K and arc_{ij}^V between elements can be added to Formula (22) and Formula (23). This improvement eliminates the need for additional linear mapping in the calculation of attention machine scores, and Formula (22) is modified to Formula (24):

$$z_i = \sum_{j=1}^n \omega_{ij}\left(x_j W^V + arc_{ij}^V\right) \tag{24}$$

Formula (23) has also been modified accordingly as follows:

$$e_{ij} = \frac{(x_i W^Q)(x_j W^K + arc_{ij}^K)^T}{\sqrt{d_z}} \tag{25}$$

due to the unique nature of positional information, the representation of relative positions varies. In order to calculate the attention scores e_{ij} of all positional pairs in a single matrix multiplication, Formula (25) is split into two parts, as shown in Formula (26):

$$e_{ij} = \frac{x_i W^Q \left(x_j W^K\right)^T + x_i W^Q \left(arc_{ij}^K\right)^T}{\sqrt{d_z}} \tag{26}$$

for the second term in Formula (26), by reshaping arc_{ij}^K to match the shape of $x_i W^Q$, the final e_{ij} can be obtained by adding the two terms. Using the same strategy, z_i can be effectively calculated, resulting in a modified Formula (27):

$$z_i = \Sigma_{j=1}^n \left(\omega_{ij}\left(x_j W^V\right) + \omega_{ij}\left(arc_{ij}^V\right)\right) \tag{27}$$

by incorporating the refined attention mechanism into the multi-head attention architecture within the Transformer model, the specific computational formulation can be delineated as follows:

$$multihead(Q, K, V) = [H_1, \ldots, H_{m_H}]W_H \tag{28}$$

$$H_h = Attention\left(QW_h^Q, KW_h^K, VW_h^V, \alpha rc^V, \alpha rc^K\right) \tag{29}$$

among them, *Attention*(.) is the self attention mechanism optimized by relative position information mentioned above. The representation of relative position relationships can be shared among multiple heads of attention, which is beneficial for the model to capture local dependencies in temporal data from multiple dimensions.

IV. EXPERIMENTS

A. DATASET AND EVALUATING INDICATOR

We validated the performance of the model through publicly available datasets and challenging industrial application scenario datasets. Firstly, the air quality dataset is used for evaluation, which focuses on time series containing multiple dynamic temporal feature inputs and target variables. Next, the performance of the model is evaluated using all complex inputs observed in industrial application scenarios, including rich static covariate features and observed time series features. The specific description of the dataset is roughly as follows:

- **Air quality dataset:** This dataset includes hourly air quality data from 12 stations in Beijing from March 2013 to February 2017. Randomly select four stations from Dongsi, Tiantan, Nongzhangguan, and Dingling, and select six temporal features: PM2.5, PM10, SO2, NO2, CO, and O3. For the missing parts of the features, the average value of the column of features is used to fill in. Before inputting into the neural network, the dataset is normalized, and there is no need for normalization when inputting into the tree model.
- **Mooney viscosity dataset:** This dataset is the actual observation data of a tire production enterprise smelting rubber in a factory. We randomly selected 10w samples with complete smelting cycles from it to evaluate the predictive performance of the algorithm. Each sample contains temporal features and static covariate features, with the prediction target being Mooney viscosity. Table 1 presents the specific characteristics of this dataset.

TABLE 1. Description of characteristic fields in the Mooney viscosity dataset.

Feature Type	Characteristic Field
temporal feature	Real time temperature.
temporal feature	Real time power.
temporal feature	Real time pressure.
temporal feature	Real time speed.
temporal feature	Real time energy.
temporal feature	Real time electricity.
static covariate feature	Total time of rubber mixing.
static covariate feature	Equipment number.
static covariate feature	Category number of rubber production.
static covariate feature	Time for discharging finished products.
static covariate feature	The starting time of the discharge door.

For all datasets, they are divided into training, validation, and testing sets in a 7:2:1 ratio. Root mean square error (RMSE) [28] and mean square error (MSE) [28] are introduced as evaluation indicators between actual and predicted values. They are represented by Formula (30) and Formula (31), respectively:

$$\text{RMSE} = \sqrt{\frac{1}{m}\Sigma_{i=1}^m \left(y_{\text{true}} - y_{\text{pred}}\right)^2} \tag{30}$$

$$\text{MAE} = \frac{1}{m}\Sigma_{i=1}^m \left|y_{\text{true}} - y_{\text{pred}}\right| \tag{31}$$

B. COMPARISON METHODS

We extensively compared AdaDynaTrans with other types of models (including tree models, transformer based temporal model variants, latest temporal models, etc.) on the public dataset. In this section, we used other models as benchmarks to evaluate the performance of AdaDynaTrans.

LightGBM [29]: LightGBM is the second mainstream Boosting algorithm implemented under the GBDT [30] framework since XGBoost [31]. LightGBM adopts a decision tree algorithm based on histograms, using GOSS single edge gradient sampling algorithm and mutually exclusive feature bundling algorithm (EOF) to improve model running speed, while supporting parallel training of class features and data, providing better parallel performance.

GRU: The traditional modeling method mainly used for time series problems is a variant of RNN. GRU introduces gating mechanisms, reset gates, and other structures to solve long-term dependency problems, making it outstanding in many time series tasks. The GRU model in this experiment consists of two layers of GRU and two fully connected layers. The hidden states of GRU are 32 and 64, respectively. The output dimension of the last fully connected layer is 1, and we set $\lambda = 0.5$.

LSTM: This model consists of two stacked LSTM layers and a fully connected layer, which extract sequence

information from input data through a recurrent network. The final fully connected layer is used to process the output of LSTM. The two LSTM layers use 128 and 64 units respectively, and a drop rate of 0.1 is used in LSTM. During the training process, use the Adam optimizer with a learning rate set to 0.025.

Transformer: This model consists of stacked encoders and decoders, with all inputs embedded and weighted with the corresponding position encoding. The encoder generates context variables, and then the decoder is followed by a fully connected layer to output the final prediction result. The setting of hyperparameters in this model (such as learning rate, etc.) is identical to the model we proposed.

STRIPE: This model represents structural diversity based on shape and time features, and introduces two differentiable and positively semi definite DPP [24] kernels to model different plans based on shape and time, ensuring both possible predictions and clear and accurate results.

AdaRNN: This model proposes the idea of time distribution partitioning for the first time from the stage of data distribution, in order to better characterize the distribution information in time series; Adopting an RNN based adaptive time prediction sequence model to reduce distribution mismatch in the time series and effectively solve the problem of time covariate shift in the time series.

C. RESULT ANALYSIS

1) AIR QUALITY PREDICTION

We employed the air quality prediction dataset to assess the efficacy of seven algorithms, including AdaDynaTrans. The predictive performance of each model was systematically evaluated by computing the RMSE and MAE between the predicted values and actual observations.

Table 2 presents the results, where $\Delta(\%)$ and in the last column represent the average increment of indicators on the GRU baseline. Our model achieved the best results on all four proposed site datasets, with AdaDyanTrans significantly outperforming other baseline models in terms of MAE and RMSE reduction.

2) MOONEY VISCOSITY PREDICTION

We compared the predictive performance of the above 7 models on the Mooney coefficient dataset, and the MAE and RMSE of each model are shown in Table 3. The results showed that AdaDynaTrans's predictive performance was significantly better than the strong baseline model AdaRNN by 7.9% in terms of MAE reduction and the strong baseline model LightGBM by 9.4% in terms of RMSE reduction.

The predictive performance of GRU is the lowest among all models, and for temporal data, it cannot handle the relative dependencies of long time series. LSTM introduces a more complex gating mechanism, which outperforms GRU in handling long-term dependencies, resulting in slightly higher accuracy than GRU. Transformer introduces position encoding and self attention mechanisms, which can better capture global dependencies and achieve better prediction results than LSTM. The prediction results of STRIPE are more accurate than those of Transformer, while the prediction performance of LGB and AdaRNN is equivalent. AdaRNN performs time distribution partitioning and matching at the data distribution level, which effectively solves the problem of time covariate shift in time series and obtains more accurate results than Transformer; LightGBM, as a tree model based on histogram strategy, is adept at mining nonlinear relationships in data and has the ability to accurately predict.

Compared with AdaRNN, AdaDynaTrans removes noise in feature input and dynamically extracts and enriches hidden feature variables that are strongly related to the target; In the time distribution matching stage, the RNN model in AdaRNN was replaced with a relative position improved transformer, resulting in a decrease of 7.9% in MAE and 14% in RMSE, respectively; Compared with the tree model LGB based on histogram strategy, the deep learning model AdaDynaTrans has made further optimization in feature input, data distribution, and other aspects, with MAE and RMSE reduced by 9.8% and 9.4% respectively.

In order to further reflect the prediction accuracy of AdaDynaTrans and clearly demonstrate the prediction error, we randomly selected 50 samples from the test set to display the prediction results of AdaDynaTrans, LGB, and AdaRNN. As shown in the results in Figure 9, there is a significant and non negligible error between the predicted values of LGB and AdaRNN and the true values, and the degree of agreement between the predicted values and the true values is lower than that of AdaDynaTrans, indicating the superiority of AdaDynaTrans's predictive performance.

3) ABLATION EXPERIMENT

In order to meticulously quantify the contributions of each architectural component proposed in our study, we systematically performed comprehensive ablation experiments on the Mooney coefficient dataset. This involved selectively removing or substituting individual components, as outlined below, from the network structure. Utilizing Transformer as the baseline, we meticulously gauged the incremental RMSE loss incurred by each modification relative to the original network structure.

DRU: We replace each DRU with a linear layer followed by an ELU.

FFU: Ablation experiments can be conducted by replacing the softmax output of Formula (5) with trainable coefficients and removing the network that generates variable selection weights. In addition, we retain the DRU in Formula (6) and maintain a similar degree of non-linear processing.

SCEM: Set all context vectors to $c_s = c_c = c_e = c_h = 0$ and connect all converted static inputs to the inputs of the time series processing module.

TVEM: Cancel the residual connection in this module, replace the LSTM encoder with a fully connected layer, and concatenate the output with the output of the static covariate processing module.

TABLE 2. Air quality prediction results.

Model	Changping		Guanyuan		Huairou		Wanliu		Δ(%)
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	
GRU	0.0545	0.0755	0.0472	0.0571	0.0398	0.0423	0.0381	0.0473	0
LSTM	0.0475	0.0600	0.0423	0.0436	0.0431	0.0536	0.0365	0.0506	-7.2
LightGBM	0.0487	0.0724	0.0426	0.0664	0.0445	0.0653	0.0289	0.0377	-1.2
Transformer	0.0367	0.0526	0.0341	0.0460	0.0325	0.0537	0.0245	0.0435	-20.39
STRIPE	0.0312	0.0432	0.0326	0.0437	0.0284	0.0363	0.0227	0.0238	-35.57
AdaRNN	0.0278	0.0507	0.0294	0.0305	0.0254	0.0355	0.0184	0.0205	-41.40
AdaDynaTrans	**0.0242**	**0.0404**	**0.0264**	**0.0315**	**0.0213**	**0.0284**	**0.0165**	**0.0181**	**-49.13**

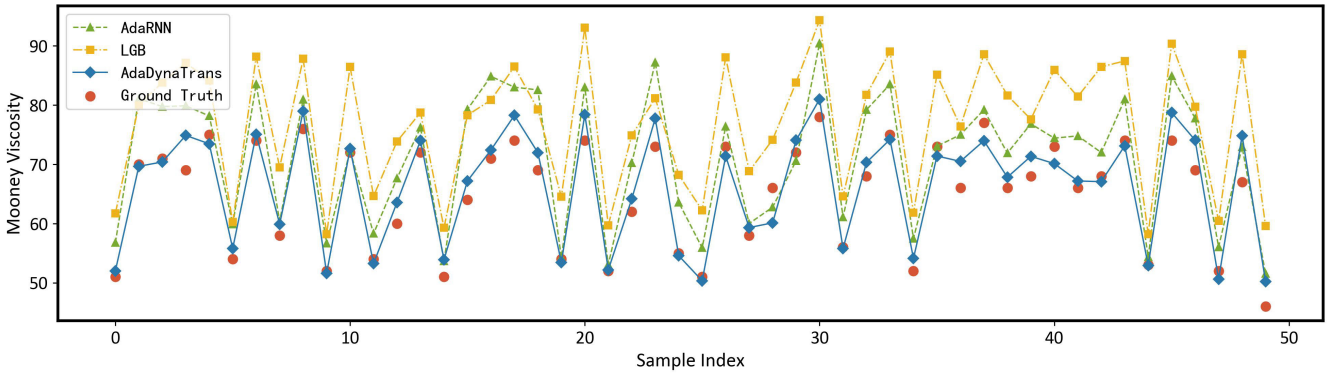


FIGURE 9. Mooney viscosity prediction results.

TABLE 3. Mooney viscosity prediction results.

Model	MAE	RMSE	$\Delta_{MAE}(\%)$	$\Delta_{RMSE}(\%)$
GRU	7.25	10.28	0	0
LightGBM	4.26	6.16	-41.2	-40.1
LSTM	6.13	8.79	-15.4	-14.5
Transformer	5.53	8.21	-23.7	-20.1
STRIPE	4.82	7.86	-33.5	-23.5
AdaRNN	4.12	6.63	-43.1	-35.5
AdaDynaTrans	**3.55**	**5.19**	**-51.0**	**-49.5**

TABLE 4. Results of ablation experiment.

Ablation Module	RMSE	$\Delta_{RMSE}(\%)$
AdaDynaTrans	5.19	-
DRU	5.41	+9.31
SCEM	5.73	+19.68
TVEM	5.98	+27.66
FFU	5.63	+16.49
TSST	5.80	+22.07
PET	5.84	+23.14

TSST: After concatenating features from SCEM and TVEM, directly input the improved relative position Transformer for TSST operation.

PET: Replace PET with a regular attention mechanism for ablation experiments in each encoder and decoder.

Table 4 shows the changes in indicators for each component after ablation testing, where the changes in indicators for each component are the average of five identical experiments taken. It can be observed that all components have a positive impact on the final prediction performance of the model, especially the time series processing module, which has the most significant impact on AdaDynaTrans. After removing TVEM, the RMSE increment is 27.66%; Secondly, the TSST used in the data distribution stage has a significant impact on the prediction performance of AdaDynaTrans, indicating that this component solves the problem of time series covariate shift to a certain extent and improves the prediction ability of the model; The results of ablation experiments on PET and the original Transformer showed that after removing relative position information in PET, the increment of RMSE was 23.14%, which had a significant impact on the ablation results of various components. This further confirmed the effectiveness of adding relative position information in PET, making it easier for the model to capture local dependencies in the time series.

V. CONCLUSION

Within this study, we introduce AdaDynaTrans, a novel multivariate time series prediction fusion model founded on the Transformer architecture. Two innovative basic units, Dynamic Residual Units (DRU) and Feature Focused Units (FFU), are conceived within AdaDynaTrans. These units constitute integral elements of time series processing modules and static covariate processing modules. Their purpose is to meticulously process data features, expunging extraneous noise while retaining essential features correlated with the predicted target. The specific processing procedure involves:

1) SCEM: This stage suppresses irrelevant information in static covariate features. Context vectors acquired from

this process are inputted into different stages of the time series processing module.

2) TVEM: This phase encompasses the processing of time series features and the fusion of contextual information through dynamic feature denoising, LSTM encoding, and final output.

In the realm of data distribution, TSST are employed to effectively characterize distribution information within the time series. An adaptive time series prediction model based on the transformer is then learned. Simultaneously, a directed graph full connection method enhances relative position information within the attention mechanism. The incorporation of a multi-head attention mechanism results in the formation of PET, further amplifying the model's performance.

AdaDynaTrans has been validated to exhibit exceptional predictive performance across both public datasets and datasets derived from actual industrial scenarios. Through comprehensive ablation experiments, we systematically validate and quantify the effectiveness and contribution of each proposed component. In the process of rubber compounding, parameters such as top ram pressure and rotor speed are manually adjusted by technicians based on monitoring curves and past experiences. Even for skilled operators, the real-time control of rubber quality in the compounding process is challenging, as operators cannot accurately predict the Mooney viscosity of rubber at a given moment by relying solely on characteristic curves. This limitation hinders the achievement of effective real-time control in rubber compounding. AdaDynaTrans addresses this challenge by enabling real-time prediction of the Mooney viscosity based on various characteristic data. This capability reduces energy and economic losses during the rubber compounding process. Using the Mooney viscosity prediction problem as a case study, this research provides insightful considerations for similar multivariate time series prediction problems.

Furthermore, our future research directions will primarily focus on two aspects. Firstly, enhancing the interpretability of the model by incorporating the ability to analyze feature importance. Secondly, leveraging the attention mechanism's capability to learn relationships between arbitrary points in time and space, our approach can be extended to model spatiotemporal data indexed by time and location coordinates. This extension will allow the construction of end-to-end systems for comprehensive modeling.

REFERENCES

[1] R. Espinosa, J. Palma, F. Jiménez, J. Kamińska, G. Sciavicco, and E. Lucena-Sánchez, "A time series forecasting based multi-criteria methodology for air quality prediction," *Appl. Soft Comput.*, vol. 113, Dec. 2021, Art. no. 107850.

[2] K. Choi, J. Yi, C. Park, and S. Yoon, "Deep learning for anomaly detection in time-series data: Review, analysis, and guidelines," *IEEE Access*, vol. 9, pp. 120043–120065, 2021.

[3] A. H. Bukhari, M. A. Z. Raja, M. Sulaiman, S. Islam, M. Shoaib, and P. Kumam, "Fractional neuro-sequential ARFIMA-LSTM for financial market forecasting," *IEEE Access*, vol. 8, pp. 71326–71338, 2020.

[4] J. Gao, H. Wang, and H. Shen, "Smartly handling renewable energy instability in supporting a cloud datacenter," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. (IPDPS)*, May 2020, pp. 769–778.

[5] Á. Briz-Redón and Á. Serrano-Aroca, "A spatio-temporal analysis for exploring the effect of temperature on COVID-19 early evolution in Spain," *Sci. Total Environ.*, vol. 728, Aug. 2020, Art. no. 138811.

[6] O. Fathi, "Time series forecasting using a hybrid ARIMA and LSTM model," Velvet Consulting, 2019, pp. 1–7.

[7] G. P. Zhang and M. Qi, "Neural network forecasting for seasonal and trend time series," *Eur. J. Oper. Res.*, vol. 160, no. 2, pp. 501–514, 2005.

[8] S. Smyl, "A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting," *Int. J. Forecasting*, vol. 36, no. 1, pp. 75–85, 2020.

[9] C. Urrea and R. Agramonte, "Kalman filter: Historical overview and review of its use in robotics 60 years after its creation," *J. Sensors*, vol. 2021, pp. 1–21, Sep. 2021.

[10] L. Guo, W. Fang, Q. Zhao, and X. Wang, "The hybrid PROPHET-SVR approach for forecasting product time series demand with seasonality," *Comput. Ind. Eng.*, vol. 161, Nov. 2021, Art. no. 107598.

[11] W. Yu, I. Y. Kim, and C. Mechefske, "Analysis of different RNN autoencoder variants for time series classification and machine prognostics," *Mech. Syst. Signal Process.*, vol. 149, Feb. 2021, Art. no. 107322.

[12] A. Yadav, C. K. Jha, and A. Sharan, "Optimizing LSTM for time series prediction in Indian stock market," *Proc. Comput. Sci.*, vol. 167, pp. 2091–2100, Jan. 2020.

[13] A. Flores, H. Tito-Chura, and V. Yana-Mamani, "Wind speed time series imputation with a bidirectional gated recurrent unit (gru) model," in *Proc. Future Technol. Conf. (FTC)*, vol. 2. Cham, Switzerland: Springer, 2022, pp. 445–458.

[14] J. Huang, S. Yang, J. Li, J. Oh, and H. Kang, "Prediction model of sparse autoencoder-based bidirectional LSTM for wastewater flow rate," *J. Supercomput.*, vol. 79, no. 4, pp. 4412–4435, Mar. 2023.

[15] Y. G. Cinar, H. Mirisaee, P. Goswami, E. Gaussier, and A. Aït-Bachir, "Period-aware content attention RNNs for time series forecasting with missing values," *Neurocomputing*, vol. 312, pp. 177–186, Oct. 2018.

[16] G. Zerveas, S. Jayaraman, D. Patel, A. Bhamidipaty, and C. Eickhoff, "A transformer-based framework for multivariate time series representation learning," in *Proc. 27th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, Aug. 2021, pp. 2114–2124.

[17] R. M. Siregar, "Mooney viscosity determination of rubber with a charm filler," *Indonesian J. Chem. Sci. Technol. (IJCST)*, vol. 4, no. 1, p. 39, Jan. 2021.

[18] H. Jin, "Ensemble just-in-time learning-based soft sensor for mooney viscosity prediction in an industrial rubber mixing process," *Adv. Polym. Technol.*, vol. 2020, pp. 1–14, Mar. 2020.

[19] W. Zheng, Y. Liu, Z. Gao, and J. Yang, "Just-in-time semi-supervised soft sensor for quality prediction in industrial rubber mixers," *Chemometric Intell. Lab. Syst.*, vol. 180, pp. 36–41, Sep. 2018.

[20] N. K. Ahmed, A. F. Atiya, N. E. Gayar, and H. El-Shishiny, "An empirical comparison of machine learning models for time series forecasting," *Econ. Rev.*, vol. 29, nos. 5–6, pp. 594–621, Aug. 2010.

[21] P. Lara-Benítez, M. Carranza-García, and J. C. Riquelme, "An experimental review on deep learning architectures for time series forecasting," *Int. J. Neural Syst.*, vol. 31, no. 3, Mar. 2021, Art. no. 2130001.

[22] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling long-and short-term temporal patterns with deep neural networks," in *Proc. Int. ACM Conf. Res. Devel. Inf. Retrieval (SIGIR)*, 2018, pp. 95–104.

[23] Y. Du, J. Wang, W. Feng, S. Pan, T. Qin, R. Xu, and C. Wang, "AdaRNN: Adaptive learning and forecasting of time series," in *Proc. 30th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2021, pp. 402–411.

[24] V. L. Guen and N. Thome, "Probabilistic time series forecasting with shape and temporal diversity," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 4427–4440.

[25] B. Lim, S. Ö. Arik, N. Loeff, and T. Pfister, "Temporal fusion transformers for interpretable multi-horizon time series forecasting," *Int. J. Forecasting*, vol. 37, no. 4, pp. 1748–1764, Oct. 2021.

[26] E. Phumnok, P. Khongprom, and S. Ratanawilai, "Preparation of natural rubber composites with high silica contents using a wet mixing process," *ACS Omega*, vol. 7, no. 10, pp. 8364–8376, Mar. 2022.

[27] Y. Zhang, H. Jin, H. Liu, B. Yang, and S. Dong, "Deep semi-supervised just-in-time learning based soft sensor for mooney viscosity estimation in industrial rubber mixing process," *Polymers*, vol. 14, no. 5, p. 1018, Mar. 2022.

[28] T. O. Hodson, "Root-mean-square error (RMSE) or mean absolute error (MAE): When to use them or not," *Geosci. Model Develop.*, vol. 15, no. 14, pp. 5481–5487, Jul. 2022.
[29] Z. Lao, D. He, Z. Wei, H. Shang, Z. Jin, J. Miao, and C. Ren, "Intelligent fault diagnosis for rail transit switch machine based on adaptive feature selection and improved LightGBM," *Eng. Failure Anal.*, vol. 148, Jun. 2023, Art. no. 107219.
[30] Q. Li, Y. Wang, Y. Shao, L. Li, and H. Hao, "A comparative study on the most effective machine learning model for blast loading prediction: From GBDT to transformer," *Eng. Struct.*, vol. 276, Feb. 2023, Art. no. 115310.
[31] A. Asselman, M. Khaldi, and S. Aammou, "Enhancing the prediction of student performance based on the machine learning XGBoost algorithm," *Interact. Learn. Environ.*, vol. 31, no. 6, pp. 3360–3379, Aug. 2023.

YINGHAO ZHU received the B.S. degree from the North China University of Science and Technology, Tangshan, China, in 2021. He is currently pursuing the master's degree with the State Key Laboratory of Industrial Control Technology, College of Control Science and Engineering, Zhejiang University, Hangzhou, Zhejiang, China.

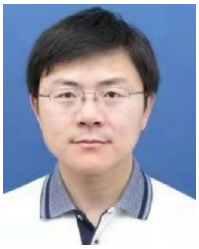
His current research interests include machine learning, big data, and artificial intelligence.

CHENHAO YING received the B.S. degree from Zhejiang University, Hangzhou, Zhejiang, China, in 2022, where he is currently pursuing the master's degree with the State Key Laboratory of Industrial Control Technology, College of Control Science and Engineering.

His current research interests include temporal forecasting and industrial intelligence.

YAN ZHOU received the B.S. degree from the University of Shanghai for Science and Technology, Shanghai, China, in 2021. She is currently pursuing the master's degree with the State Key Laboratory of Industrial Control Technology, College of Control Science and Engineering, Zhejiang University, Hangzhou, Zhejiang, China.

Her current research interests include computer vision, artificial intelligence, and medical imaging.



JIANGANG LU received the B.S. and Ph.D. degrees from Zhejiang University, Hangzhou, China, in 1989 and 1995, respectively.

He is currently a Full Professor with the Institute of Industrial Process Control and the College of Control Science and Engineering, Zhejiang University. He is also affiliated with the State Key Laboratory of Industrial Control Technology, College of Control Science and Engineering, Zhejiang University. He has published one book and more than 120 papers in journals and conferences. His research interests include artificial intelligence in industries, intelligent sense, intelligent modeling, intelligent control, intelligent optimization, and industrial big data. He is a member of the Technical Committee on Process Control in the Chinese Association of Automation and the Standing Director of the Zhejiang Association of Automation. He received the First Prize of China's National Science and Technology Progress Award, in 2013, the Geneva International Invention Gold Medal Award, in 2013, and the Second Prizes of Zhejiang Province Science and Technology Progress Award, in 1999 and 2007.