IEEE Access
Multidisciplinary ⁝ Rapid Review ⁝ Open Access Journal

Received 12 June 2024, accepted 10 August 2024, date of publication 15 August 2024, date of current version 26 August 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3443876

APPLIED RESEARCH

A Time-Series Model of Gated Recurrent Units Based on Attention Mechanism for Short-Term Load Forecasting

WUTAO XIONG
College of Electronics and Information Engineering, Sichuan University, Chengdu 610065, China
e-mail: 674820272@qq.com

ABSTRACT Accurate short-term load data is foundational for rigorous research and policy formulation. We propose a machine learning framework capable of precisely forecasting future loads and addressing missing data to meet this requirement. This framework integrates considerations of temporal autocorrelation and inter-feature correlations, ensuring a high degree of generalizability across diverse load datasets. Our investigation assesses the efficacy of several machine learning architectures, including convolutional neural networks and gated recurrent units, in solving this problem. We also introduce a neural network-based time-series model to extract and leverage temporal and inter-feature correlations within these datasets. The proposed models are subjected to rigorous experimentation and validation using a real-world dataset to ascertain their robustness. This study aims to offer robust and efficient methodologies for analyzing electricity consumption data in both academic and practical contexts.

INDEX TERMS Attention, residual-convolution neural network, load forecasting, gate recurrent unit.

I. INTRODUCTION

Partial data loss is a common issue faced by power consumption datasets. However, data completeness is crucial when evaluating parameters such as demand, voltage, and current [1].

Thus, exploring methods for filling in the missing data is essential. As the data is only partially missing, we can transform the filling problem into a prediction problem. Additionally, we may want to predict future data, so we should combine filling and prediction into a single prediction problem.

A. RESEARCH STATUS

The primary methodologies employed to tackle these issues encompass mathematical and statistical techniques and machine learning algorithms.

1) TRADITIONAL METHODS

Statistical methods such as exponential smoothing [2], Kalman filtering [3], Non-Linear Mixed-Effects Model [4], and Semi-Parametric additive models [5] are commonly used for load forecasting at an aggregate level, but are not directly applicable for individual households or blocks (the minimum statistical unit). Moreover, these methods typically provide one-dimensional predictions based on limited input features. Specifically, they primarily rely on time-lagged time-series data to predict missing values without considering factors such as weather and holidays that may impact electricity consumption on the day in question. Additionally, these methods often overlook nonlinear relationships between different features on the same day. An alternative approach is to use other features available simultaneously to predict the missing values, ignoring the temporal changes in the data and focusing solely on the relationships between the features. However, this approach fails to consider changes in the features over time and misses out on potential correlations among temporal data.

Machine learning methods can be divided into traditional methods, such as Support Vector Machine (SVM) [6], Shallow Neural Networks (SNN) [7], K-Nearest Neighbors (KNN) [8], and Random Forest (RF) [9]. However, due to their design principles, these methods could improve their ability to effectively extract the temporal correlations

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

in time series data [1], resulting in poor performance in load forecasting tasks [10]. Previous studies have attempted to improve the performance of traditional methods by combining multiple models, such as a novel model that integrates wavelet transform (WT), grey model (GM), and particle swarm optimization (PSO) [11], or the conditional restricted Boltzmann machine (CRBM) and factorized CRBM (FCRBM) derived from artificial neural network (ANN) prediction methods [12]. Hybrid methods have shown better performance than single traditional methods, reducing prediction errors. However, these methods still exhibit poor long-term memory when dealing with load time series data.

2) DEEP LEARNING METHODS

Compared to traditional models, deep learning models have generally shown better performance in exploring the non-linear relationships in complex problems [13]. Among them, Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN) have shown significant effectiveness in time series processing [14], [15], such as the sequence-to-sequence (seq2seq) processing based on Long Short-Term Memory (LSTM) models [16], and the optimized Gated Recurrent Unit (GRU) 47 models for LSTM [17].

Research has also been conducted on online learning and dynamic neural network models [14], [18]. Such as an online adaptive RNN that can continue learning while receiving data [14], and the development of a dynamic neural network load model for power system reliability assessment based on dynamic loads [19].

In addition, some mechanisms in the Transformer model have also demonstrated exemplary performance in processing temporal content [20]. However, due to its strict requirements for the dataset and the high computational demands, its application is limited.

Therefore, despite many previous studies on short-term load forecasting (STLF), models or methods have ignored the autocorrelation of load over time or the intercorrelation of different features simultaneously. Or they have been too demanding regarding data and computing requirements [20] to explore the overall correlation between the data better. However, filling in missing data requires paying as much attention as possible to this type of information. Therefore, exploring data features from two directions of time and features and reflecting the "two-dimensional characteristics" of missing values as much as possible is a feasible solution to address the existing limitations.

B. OUR METHOD

To overcome the abovementioned challenges, the present study introduces an innovative deep learning framework that integrates a Residual-CNN, a GRU-based Seq2Seq model, and an attention module in conjunction with PCA for data volume reduction. This architecture facilitates the prediction of missing values by simultaneously exploring the interplay between temporal sequences and various features. Imputing missing values is redefined as a multivariate time series forecasting issue, wherein the series comprises samples at consistent time intervals. The model's inputs are the loads for each segment, which include prior loads for each segment and the current segment's non-missing data. The CNN employs one-dimensional convolutions along the temporal axis to capture spatial characteristics associated with different variables at each discrete time segment. Subsequently, these spatial features are fed into the Seq2Seq network composed of a dual-layer GRU structure (comprising both decoder and encoder) to deduce temporal attributes from the time dimensionality. An attention strategy akin to the one implemented in Transformers is crafted to refine the Seq2Seq's capability to access information, enabling the network to consider every segment of data, thereby yielding a more consistent forecast trajectory.

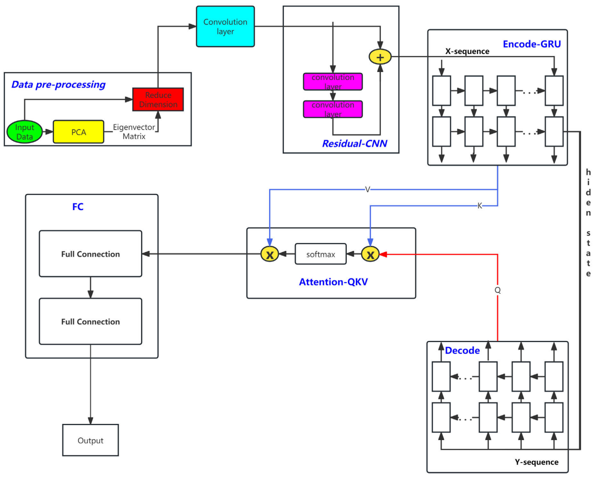

FIGURE 1. The whole structure.

C. MAIN CONTRIBUTION

The contributions lie in three folds:

- A hybrid architecture model combining residual Residual-CNN, GRU, and attention mechanism, with the use of PCA for data preprocessing, is devised for short-term load forecasting (STLF). This model has the ability to extract information from both spatial and temporal dimensions simultaneously.
- The use of attention mechanism is extended from image and natural language processing to data processing.
- Multiple models were compared and evaluated in terms of accuracy, time consumption, and smoothness. The proposed model achieved good prediction performance while minimizing the computational requirements.

D. MAIN CONTENT AND STRUCTURE

The structure of this thesis consists of six chapters, arranged as follows:

- Chapter One functions as an introduction, commencing with an examination of the research context and the imperative nature of short-term load forecasting. It further discusses the current state of research on short-term load forecasting, highlighting the limitations of existing methods. The chapter concludes by presenting the research approach of this thesis, briefly describing the model architecture, and delineating the main content and innovative aspects of this study.

- Chapter Two introduces the theoretical background related to the work, primarily discussing knowledge pertaining to Seq2Seq(sequence to sequence).
- Chapter Three analyzes a time-series model employing Gated Recurrent Units (GRUs) integrated with an attention mechanism for short-term load forecasting. Additionally, the chapter outlines the implementation details of the proposed model.
- Chapter Four conducts experiments on publicly available datasets. Compared with existing methods, the results validate the efficacy of this approach. It also includes ablation studies to explore the impact of different modules. Finally, the chapter utilizes multiple evaluation methods to assess the outcomes of model training and prediction.
- Chapter Five concludes the thesis, evaluating the model's results and identifying current shortcomings. It concludes with plans for future work.

E. SYMBOL DEFINITION

TABLE 1. Symbol Definition.

Symbol	Definition
Z_{ij}	The standardized matrix calculated using the Z-score method
S_j	The j-th column's standard deviation in the original dataset
r_{ij}	The correlation matrix's element located at the intersection of the i-th row and j-th column represents the correlation coefficient.
$\mathbf{R}$	The correlation coefficient matrix
l_{gn}	The n-th dimensional feature of eigenvector
$\mathbf{L}_g$	The eigenvector corresponding to the eigenroot
$\mathbf{P}$	k*n dimensional PCA eigenmatrix for dimensionality reduction
$\mathbf{ReLU}$	ReLU activation function
$\sigma()$	sigma activation function
r_t	The output of the reset gate function in Gated Recurrent Units (GRUs), regulating the quantity of historical information transferred to the current state
z_t	The output of the update gate function in Gated Recurrent Units (GRUs), regulating the quantity of previous time information transferred to the current state
$\tanh$	tanh activation function
$\tilde{\mathbf{he}}_t$	The memory content of the current GRU unit
$\mathbf{ye}_t$	The encoder module output of the current GRU unit
$\mathbf{he}_t$	The output hidden state of the present GRU cell / the input hidden state of the subsequent cell
$\mathbf{hd}_t$	The input of the t-1 unit of the decoder module and the output of the t unit. Especially, $hd_0 = he_n$ (i.e. the final state of the encoder as input to the decoder)
$\mathbf{yd}_t$	Decoder output of the current GRU unit
$\mathrm{Softmax}(x)$	Normalized exponential function
m_{ij}	Product of decoder and encoder output
$\mathbf{M}_{ij}$	The vector formed by $m_{i,j}$

II. BACKGROUND: SEQ2SEQ

The inception of the Seq2Seq modelling approach was marked by foundational research conducted by Sutskever et al. [21] and Cho et al. [22], who demonstrated its efficacy in machine learning.

The sequence-to-sequence (Seq2Seq) framework is pivotal in the domain of machine learning for converting a sequence from one domain to an analogous sequence in another domain. This paradigm facilitates the conversion of textual information from one linguistic form to another. It is important to note that this language is not just a natural language.

The encoder's role is to process the input sequence and condense its information into a compact form, commonly referred to as the 'thought vector.'(Final hidden vector of the encoder). This vector endeavours to encapsulate the quintessential aspects of the input. Subsequently, the decoder undertakes the responsibility of interpreting this vector to construct the desired target sequence.

In the realm of machine translation, for example, the encoder ingests a sentence articulated in the source language while the decoder strives to produce a semantically equivalent translation in the target language. This sophisticated architecture is adept at managing sequences of variable lengths, an attribute critical to the intricate nature of human languages.

Seq2Seq architectures frequently leverage the capabilities of recurrent neural networks (RNNs), with Long Short-Term Memory (LSTM) networks or Gated Recurrent Units (GRUs) being particularly prevalent due to their efficacy in preserving information across sequences. Innovations in this area include the integration of attention mechanisms, which empower the model to focus dynamically on pertinent segments of the input sequence during the generation of each successive element of the output sequence. Furthermore, Transformer models represent a paradigm shift, eschewing conventional RNN structures in favour of pervasive attention-based mechanisms.

We show the architecture of the common RNN in Fig.3, LSTM in Fig.4 and Transformer unit in Fig.5, and Seq2Seq encoder-decoder in Fig.2 without too much explanation. The GRU units that will be used will be explained in detail below.

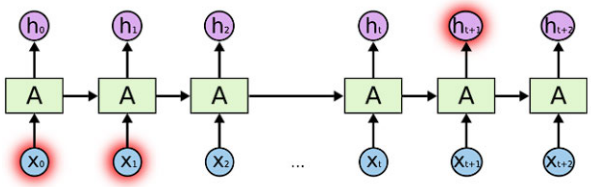

FIGURE 2. Seq2Seq Encoder-Decoder [23].

All the units shown below can be placed at "A" in Fig.2. Perform encoding and decoding work. But it's important to note that the transformer unit is unique. The left side of the unit is what its encoder uses, and the right side is what its decoder uses. The Encoder-Decoder form of this unit is to expand the left and right sides n times respectively, as shown in the figure "N×".

III. METHODOLOGY AND MODEL

A. PRINCIPLE COMPONENT ANALYSIS

Given the extensive number of features in the initial dataset, utilizing the data directly for training leads to suboptimal performance. Thus, it is crucial to implement suitable data

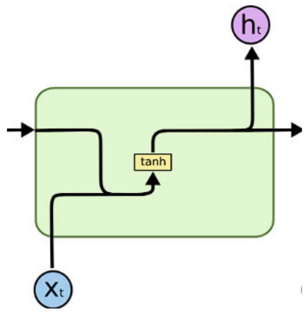

FIGURE 3. Common RNN unit [23].

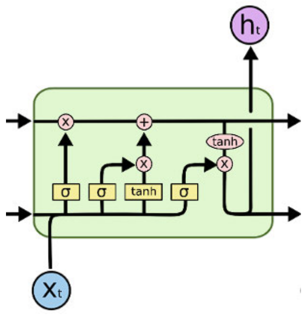

FIGURE 4. LSTM unit [23].

processing techniques to decrease the data's dimensionality. Reducing dimensionality offers multiple benefits, such as simplifying the dataset, lowering computational demands, eliminating noise, and clarifying the outcomes [24].

Principal Component Analysis (PCA) stands as the preeminent technique for reducing the dimensions of data. The core concept of PCA is to project features from an n-dimensional space into a k-dimensional subspace, where these k-dimensions comprise a novel assortment of orthogonal attributes, termed principal components, derived from the initial n-dimensional space. Within the scope of this research, PCA is employed to compress the dimensionality of 320 distinct power consumption attributes down to 50 dimensions, thereby significantly enhancing computational efficiency and diminishing the influence of noise.

Suppose the jth dimension feature value of the ith time point in the original data is denoted by x_{ij}, then the original data can be viewed as $m \times n$ matrix:

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} & \cdots & x_{1n} \\ x_{21} & x_{22} & x_{23} & \cdots & x_{2n} \\ x_{31} & x_{32} & x_{33} & \cdots & x_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & x_{m3} & \cdots & x_{mn} \end{bmatrix}_{m\times n}$$

To standardize the data on the indicators, we used the Z-score method to normalize the sample matrix. The normalized matrix, denoted as Eq.(1), is:

$$Z_{ij} = \frac{x_{ij} - \overline{x}_j}{S_j} \tag{1}$$

$$S_j = \sqrt{\frac{\sum_{i=1}^{n}(x_{ij} - \overline{x}_j)^2}{n-1}} \tag{2}$$

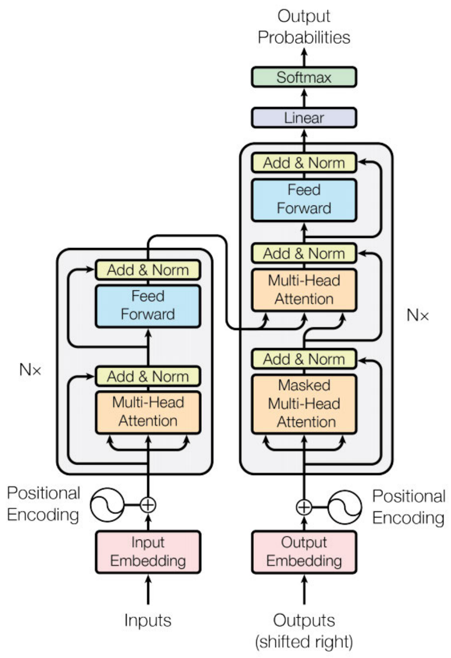

FIGURE 5. Transformer unit [20].

where Eq.(2) states that S_j is the standard deviation of the j_{th} dimension data. Based on the normalized matrix Eq.(1), we can obtain the correlation coefficient matrix R as Eq.(4):

$$r_{ij} = \frac{\sum_{i=1}^{n} Z_{ij}Z_{ij}}{n-1} \tag{3}$$

$$\mathbf{R} = (r_{ij})_{m\times m} \tag{4}$$

For the feature equation, we can find the eigenvalues $\lambda_1, \lambda_2 \cdots, \lambda_m$(arranged in descending order). They represent the variance of each principal component and describe the magnitude of each principal component. Each eigenvalue corresponds to an eigenvector, as shown in Eq.(5).

$$\mathbf{L}_g^T = [l_{g1}, l_{g2}, \ldots, l_{gm}], \qquad g = 1, 2, \ldots, m \tag{5}$$

Since the eigenvalues are arranged in descending order, the top k values can be chosen. The corresponding k eigenvectors are then selected as row vectors to create a feature vector matrix, as illustrated in Eq.(6). The data undergoes transformation into a novel space delineated by the k eigenvectors, as shown in Eq.(7).

$$\mathbf{P} = \begin{bmatrix} l_{11} & l_{12} & \cdots & l_{1m} \\ l_{21} & l_{22} & \cdots & l_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ l_{k1} & l_{k2} & \cdots & l_{km} \end{bmatrix}_{k\times m} \tag{6}$$

$$\mathbf{Y} = \mathbf{PX} \tag{7}$$

B. RESIDUAL-CONVOLUTION NEURAL NETWORK

Within the structure of a CNN, a standard convolutional block is generally composed of three essential layers: A convolutional layer performs convolution operations, a linear mapping activation layer, and a pooling layer that reduces the size of the parameter matrix. The convolutional layer manages the initial processing of the input data and forwards the output to the activation function. The pooling layer subsequently processes this output. However, in our model, pooling is unnecessary; therefore, we did not use any pooling layer. The activation layer (activation function) used in our experiments is Rectified Linear Units (ReLU) [3], [9], which ensures smoother training and better convergence of learning curves. Additionally, we employed residual connections (ResNet) to ensure the effectiveness of the training. We refer to this design as Residual-CNN (RCNN). Its architecture is shown in Fig.6

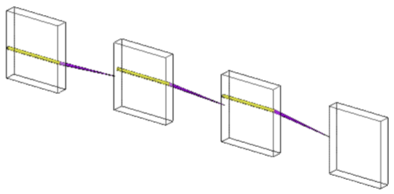

FIGURE 6. **Convolutional module schematic diagram.**

After PCA processing, the dimensionality of the data is reduced to 50 dimensions. RCNN receives these 50-time series variables as inputs, and to reduce the computational complexity during training, we slice the data into segments of 60-time steps each. One-dimensional convolution is then applied to extract temporal features from the first 60 time steps of each segment. In each convolutional layer of the CNN, the input data is processed by the convolutional layer and the resulting output is passed through an activation function. Eq.(8) describes the output of the convolutional layer for the input sample y_{ij}:

$$a_{ij} = \sum_{k=1}^{K} y_{kj} w_{i+m-1} + b_j \tag{8}$$

In Eq.(8), the term y_{kj} represents the input vector at the j^{th} timestep, b_j refers to the bias, w signifies the weight associated with the convolutional kernel of length K, and a_{ij} denotes the output produced prior to activation.

The Rectified Linear Unit (ReLU) function sets the output of some neurons to zero, which can induce sparsity in the network and reduce the interdependence of parameters, thereby alleviating overfitting. Eq.(9) outlines the output of the ReLU layer, with ReLU() denoting the activation function and z_{ij} representing the resulting output.

$$z_{ij} = \mathrm{ReLU}(a_{ij}) \tag{9}$$

$$\mathrm{ReLU}(x) = \max(0, x) \tag{10}$$

This convolution operation is repeated four times in the model. Additionally, a residual connection is performed after every two convolutional layers to improve training. This operation is described by Eq.(11).

$$\mathrm{H}(x) = \mathrm{F}(x) + x \tag{11}$$

where x represents the feature input to the convolutional layer, $\mathrm{F}(x)$ represents the feature output after two convolutions, and $\mathrm{H}(x)$ represents the output after the residual connection.

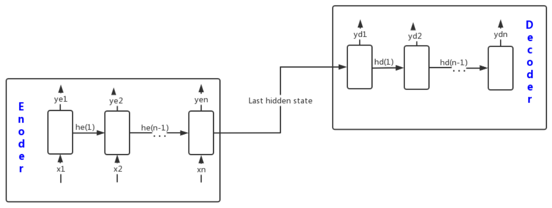

FIGURE 7. **The principle of Seq2Seq network.**

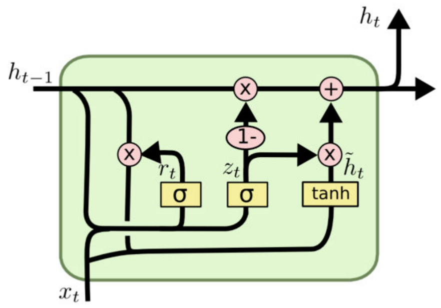

FIGURE 8. **The principle of GRU [23].**

C. GRU SEQUENCE-TO-SEQUECE NETWORK

The time series data for the same power load exhibit significant temporal correlation. The primary objective of the GRU-Seq2Seq network is to enable the model to concentrate on data from earlier, more remote time intervals, thereby enhancing the model's memory capacity and overcoming the constraint of merely concentrating on local fluctuations.

The network consists of two interconnected GRU layers in a Seq2Seq architecture, forming the Encoder module and the Decoder module, respectively. The architecture is depicted in Fig.6. The Encoder module's endpoint is linked to the Decoder module's starting point, with the Encoder's hidden variable output transmitted to the Decoder. These two connected modules together constitute the Encoder-Decoder module.

The GRU in the encoder contains two gates, the reset gate and the update gate, which regulate the information flow between them. Each unit has two inputs: the prior hidden state and the current input at that time step, along with one output, the hidden state that will be sent to the following unit.

The reset gate (Eq. (13)) acts on the previous hidden state and determines what past information should be forgotten. The update gate (Eq.(14)) acts on the current and previous hidden states, determining what information should be passed down. The new hidden state is then updated according to Eqs. (16-17). After these transformations, h_t is inputted to the next unit or transformed into the output ye_t according to Eq.(18).

Its overall architecture is shown in Fig.8.

$$\sigma_{ei} = \frac{1}{1 + e^{-\mathbf{x}}} \tag{12}$$

$$\mathbf{r_t} = \sigma_{e1}\left(\mathbf{W}_r \cdot [\mathbf{he}_{t-1}, \mathbf{x_t}]\right) \tag{13}$$

$$\mathbf{z_t} = \sigma_{e2}\left(\mathbf{W}_z \cdot [\mathbf{he}_{t-1}, \mathbf{x_t}]\right) \tag{14}$$

$$\tanh(x) = \frac{e^{\mathbf{x}} - e^{-\mathbf{x}}}{e^{\mathbf{x}} + e^{-\mathbf{x}}} \tag{15}$$

$$\tilde{\mathbf{he}}_t = \tanh\left(\mathbf{W}_{\tilde{\mathbf{he}}_t} \cdot [r_t \cdot \mathbf{he}_{t-1}, \mathbf{x_t}]\right) \quad (16)$$

$$\mathbf{he}_t = \mathbf{z_t} * \tilde{\mathbf{he}}_t + (1 - \mathbf{z_t}) * \mathbf{he}_{t-1} \quad (17)$$

$$\mathbf{ye}_t = \sigma_{e3}(\mathbf{V} \cdot \mathbf{he}_t) \quad (18)$$

where * denotes the Hadamard product.

The decoder unit in the GRU-Seq2Seq network takes the last hidden state of the encoder unit as its initial input (Eq. (19)). This state contains the historical information of the sequence. The unit generates a hidden state based on the input, which follows Eq. (20). The hidden state is passed to the next unit as input and also fed into the output module, which produces yd_t according to Eq. (21).

$$\mathbf{hd}_0 = he_n \quad (n \text{ is the length of } \mathbf{he}) \quad (19)$$

$$\mathbf{hd}_t = \sigma_{d1}(\mathbf{U} \cdot \mathbf{hd}_{t-1}) \quad (20)$$

$$\mathbf{yd}_t = \sigma_{d2}(\mathbf{V} \cdot \mathbf{hd}_t) \quad (21)$$

The output of the Encoder-Decoder module will be passed to the Attention part.

D. ATTENTION MECHANISM

The Attention mechanism [20] is shown in Fig.9, which has a solid ability to extract information from data at a global level. Using this mechanism can effectively smooth out prediction/filling curves, reducing the risk of model overfitting and minimizing drastic local changes.

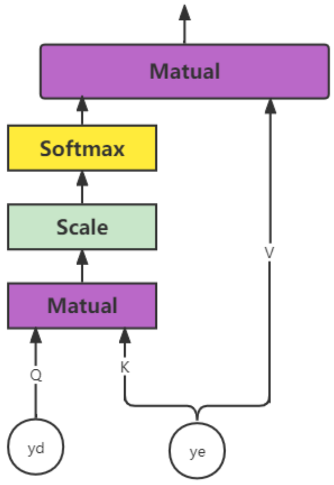

FIGURE 9. The principle of attention mechanism.

The Attention mechanism can be summarized by the mathematical relationship Eq.(22), where the softmax function is shown in Eq.(23). The process splits the input into query (Q), key (K), and value (V) components. It assesses the relationship between Q and K to decide which V to retrieve. The mechanism uses a dot product to calculate similarity and then divides by the square root of the dimension to unify the data scale. The softmax layer is used to probabilize the values, and the probability is multiplied by V to extract essential features [20]. In the model, the input values have consistent dimensions due to previous processing, simplifying the uniform scale operation.

In this model, we consider ye as Q and V and yd as K. Therefore, we calculate it based on Eq. (22) and provide Eqs. (24-28) for calculation, obtaining a vector expression focusing on the entire domain. This expression is then inputted into the next part.

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{\mathbf{QK}^T}{\sqrt{d_k}}\right)\mathbf{V} \quad (22)$$

$$\text{Softmax}(x) = \frac{e^{x_i}}{\sum_i e_i^x} \quad (23)$$

$$m_{ij} = \mathbf{ye}_i \cdot \mathbf{yd}_i \quad (24)$$

$$M_{Ij} = m_{1j}, m_{2j} \cdots m_{Ij} \quad (25)$$

$$P_{Ij} = \text{Softmax}(Y_{Ij}) \quad (26)$$

$$Y_I = \{\mathbf{ye}_1, \mathbf{ye}_2, \ldots, \mathbf{ye}_I\} \quad (27)$$

$$\text{Attention\_model}(\mathbf{ye}, \mathbf{yd}, \mathbf{ye}) = \sum_j \mathbf{P}_{Ij}\mathbf{Y}_I \quad (28)$$

E. FULLY CONNECTED NETWORK

To make predictions for a future time period, the output data from the Attention Mechanism needs to pass through a fully connected layer. The computation performed by this layer is represented by Eq.(29).

$$p_{it} = \text{ReLU}\left(\sum_{k=1}^{K} w_k x_k + b_i\right) \quad (29)$$

In Eq.(29), p_{it} signifies the forecast for time t, *ReLUs* denotes the ReLU activation function, w and x represent the weight and associated feature vector, respectively, b indicates the bias, and k is the count of nodes within the fully connected layer.

IV. RESULT

In this section, we performed a comparative analysis of our proposed model against several commonly used architectures using the UCI Electrical Power dataset [25]. We specifically assessed our model's performance in the following aspects:

- Analyze the performance of different modules (CNN, Encode-Decode, and Attention) by disassembling and testing them separately.
- Performance of filling missing data at different time periods.
- Accuracy of missing information imputation (prediction).
- We also conducted repeated experiments on other datasets and found no significant differences in predictive performance and cost among the various models. Therefore, this part was not further elaborated.

A. EVALUATION METHOD

In this experiment, we employed various deep-learning approaches to predict or fill in missing data in the future. For filling in missing data, we allowed the model to use lagged inputs of the feature for prediction, while for prediction, we treated it as filling in missing data on the time series.

We used the first 70% of the time period as training data and the last 30% as testing data. We evaluated the model's prediction using the mean squared error (Eq. (30)) as the loss function. We note that some studies also used the root mean squared error (Eq. (31)) for evaluation [1]. However, MSE and RMSE only differ in the scale of the data. Therefore, we only used the MSE loss function.

$$\mathrm{MSE} = \frac{1}{M}\sum_{i=1}^{M}(y_i - \hat{y}_i)^2 \tag{30}$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{M}\sum_{i=1}^{M}(y_i - \hat{y}_i)^2} \tag{31}$$

where y_i represents the i_{th} predicted value by the model, while y_i represents the corresponding actual value. M represents the number of training samples.

In addition, the smoothness of the predicted curve is an important indicator that reflects the model's ability to consider the global trend. Therefore, we also set up a smoothness score function. Since the time difference is fixed, the difference between two predicted values can reflect the degree of variation. The calculation formula evaluates the smoothness level by computing the sample standard deviation (Eq.(33)) of the predicted data differences (Eq.(32)). The calculation method for the smoothness score is shown in Eq.(34).

$$\mathrm{Diff}_i = x_{i+1} - x_i \tag{32}$$

$$\mathrm{STD}(x) = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2} \tag{33}$$

$$\mathrm{Smooth\_Score} = \mathrm{STD}(\mathrm{Diff}) \tag{34}$$

where x_i represents the i_{th} predicted value, $\overline{x}$ represents the mean value of the predicted values in the group, and N represents the number of predicted values in the group.

B. DATASET ANALYSIS

We first analyze the dataset we used:

1) DATASET PREPROCESSING

The challenge of electricity load forecasting lies in capturing the spatiotemporal relationships of the data. In this study, we utilized the electricity dataset from UCI, which contains the electricity consumption (in kWh) of 320 customers every 15 minutes from 2011 to 2014. For some customers, their electricity consumption data was recorded starting from a different time, and the data before the first record was marked as 0.

As our trained model requires a starting point for the time series, and different customers have different starting times, some features remain 0 before a certain time point. Using all the data for training would inevitably have a negative impact. Therefore, we removed the data from 2011 and set the starting time to January 1, 2012. Additionally, we converted the data into hourly consumption using mathematical methods to extract features and reduce the data volume. Hence, the preprocessed data contains the hourly electricity consumption of each object from 2012 to 2014.

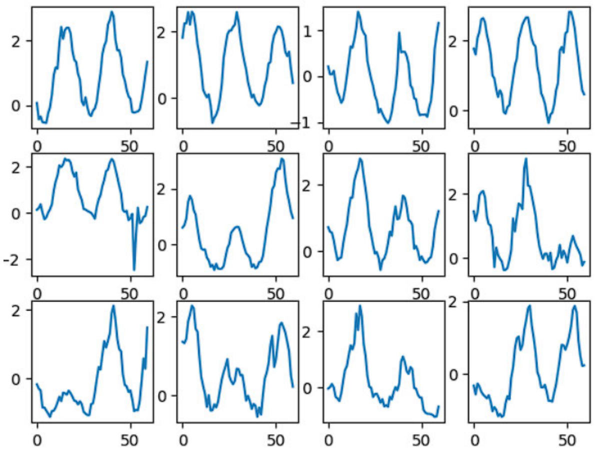

FIGURE 10. Image of the preprocessed data.

2) DATASET VISUALIZATION

We'll plot the preprocessed data using Python code. Since the data is too long, we display the data in slices. The visualization results are shown in the Fig.10.

C. SEED SELECTION

The initial seed configuration markedly affects the ultimate outcomes of model training [26], making it crucial to investigate how various seeds influence the model's performance. We randomly tried one hundred random seeds, but only two images(Fig.11 and Fig.12) are shown here for comparison. The remaining pictures are in the Appendix.

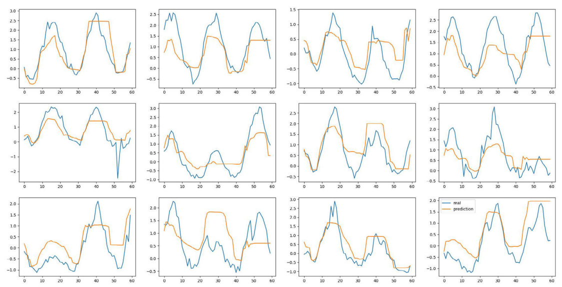

FIGURE 11. Result when seed=8.

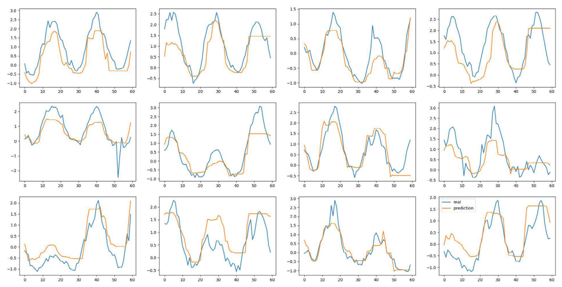

FIGURE 12. Result when seed=16.

At the same time, we calculate the difference between the evaluation parameters of the training results, which is used to prove the influence of the seed selection.

The data in Table 2 demonstrate that the model's Mean Squared Error (MSE) varies between 0.0667 and 0.113. The multiples obtained were 2.71 times and 1.56 times, indicating that the choice of seed significantly impacts the results.

TABLE 2. Comparison of MSE degree among different seeds.

MSE	Train_loss	Validation_lossALL
Average	0.0692	0.239
Std	0.000156	0.000338
Worst	0.105	0.316
Best	0.0389	0.203

D. PERFORMANCE ANALYSIS OF PREDICTION

1) PREDICTION ACCURACY

Table.3 presents the mean squared error (MSE) (Eq. (27)) of three models, which are shown in:

- Fig.13: The model in paper(ATTENTION)
- Fig.14: those without the attention mechanism but with gated recurrent units (GRU)
- Fig.15: those only using convolutional neural networks (CNN).

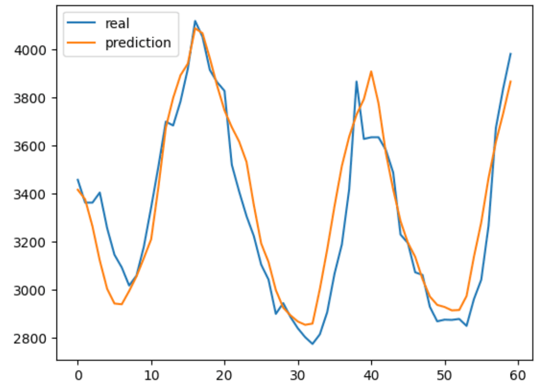

FIGURE 13. ATTENTION.

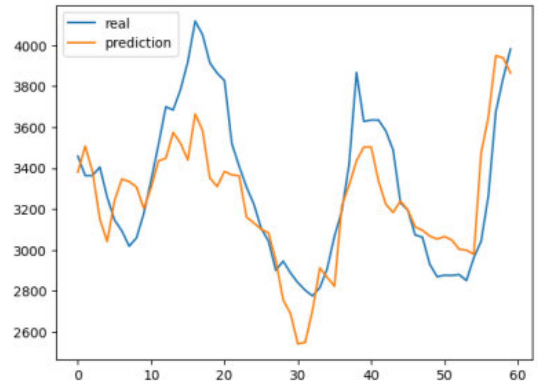

FIGURE 14. GRU.

The model using the attention mechanism achieves the best prediction performance, with an average improvement in prediction/filling accuracy of 2.6% and 15.1% compared to GRU and CNN, respectively. The optimal performance of the model is improved by 1.8% and 11.1%, respectively. Furthermore, the standard deviation of the model using the attention mechanism is smaller than that of GRU and CNN, indicating more consistent training performance.

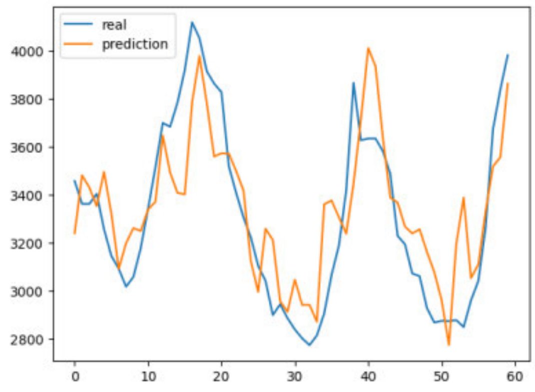

FIGURE 15. CNN.

TABLE 3. Comparison of Prediction Accuracy Among Different Modules.

MSE	ATTENTION	GRU	CNN
Average	0.185	0.190	0.218
Best	0.168	0.171	0.189
Std	0.010	0.012	0.015

2) SMOOTHING DEGREE

Table.4 shows that the smooth score (Eq. (34)) of the model with the ATTENTION mechanism is significantly better than that of the models without this mechanism. The model with the ATTENTION mechanism has the best smoothing effect, with an average score better than the best of other models, demonstrating the global information mining ability of this mechanism [21]. Its average smooth score is increased by 22.3% and 40.8% compared to GRU and CNN, respectively. Its best performance is improved by 20.8% and 39.1%, respectively. Moreover, its standard deviation is lower than that of GRU and CNN, suggesting more stable training performance across the model.

TABLE 4. Comparison of curve smoothing degree among different modules.

Smooth Score	ATTENTION	GRU	CNN
Average	0.171	0.220	0.289
Best	0.156	0.197	0.256
Std	0.008	0.009	0.017

3) ALGORITHM TIME COST ANALYSIS

Table.5 shows that the model with the ATTENTION mechanism has a longer runtime than those without this mechanism. Considering that the model's training time and computation are linearly related, the runtime can be used as a proxy for computational complexity. Although the use of

TABLE 5. Comparison of time cost among different modules.

Time Cost	ATTENTION	GRU	CNN
	(second)		
Average	18.397	13.869	11.381
Best	16.085	10.635	8.135
Std	1.395	2.477	1.903

the ATTENTION mechanism incurs higher computational costs than CNN and GRU, the improvement in smoothing and prediction accuracy can offset the computational loss. Moreover, the model's computational cost is much lower than the Transformer's.

The model with the ATTENTION mechanism has the best smoothing performance, with an average score that exceeds the optimal score of other models, demonstrating the mechanism's ability to capture global information. Its average smooth score is 22.3% and 40.8% higher than GRU and CNN, respectively, while the best performance is 20.8% and 39.1% higher. Additionally, its standard deviation is smaller than that of GRU and CNN, indicating a more consistent model training performance.

E. ABLATION STUDY

Our experiments primarily involve three modules: Attention, GRU, and CNN, designated as Module A (Attention), Module B (GRU), and Module C (CNN), respectively. The experimental design includes scenarios with only one module, combinations of two modules, and all three modules together. We have three configurations for individual modules: A, B, and C. Combinations of two modules include various pairings as depicted in the accompanying Table.6. It should be noted that the sequence of combination maintains uniqueness; that is, B+A is equivalent to A+B. Furthermore, the configuration involving all modules is labelled A+B+C, resulting in seven distinct combinations. Importantly, the model architecture concludes with a fully connected layer in any combination.

TABLE 6. Combination of modules.

Module 2 / Module 1	**A**	**B**	**C**
A	╲	A+B	A+C
B	B+A	╲	B+C
C	C+A	C+B	╲

In this set of experiments, uniform initial parameters, specifically the random seed suffices, are ensured. The models are not optimized to highlight the comparative strengths and weaknesses via figure.

The performance of individual modules A, B, and C is significantly inferior compared to their combinations;

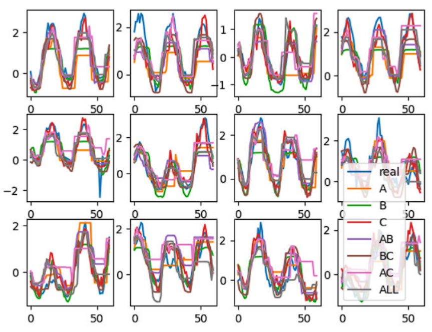

FIGURE 16. The principle of attention mechanism.

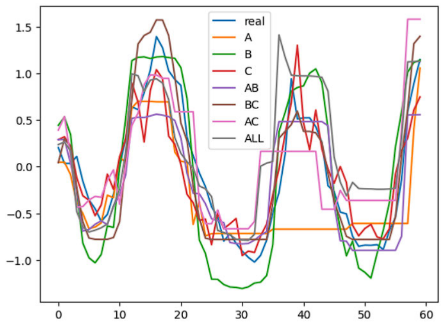

FIGURE 17. The principle of attention mechanism.

TABLE 7. Comparison of prediction accuracy among different modules.

Score / Model	Train_Loss	Validation_loss	Smooth_Score
A	0.167792842	0.347791672	0.982355952
B	0.997972906	0.844338477	1.058599949
C	0.067561217	0.260044277	0.991613269
AB	0.089239784	0.294330776	1.354439735
BC	0.365758419	0.347687244	1.022817731
AC	0.365758419	0.347687244	1.022817731
ALL	0.068685919	0.18923384	0.873270714

therefore, they are not prominently featured in the significant comparison graphs but are shown in the broader comparative diagrams through their plotted curves. Consequently, we draw Fig.16 that contains all the predicted curves, and Fig.17 that shows only partial information.

We similarly employ "Train_Loss", "Validation_loss", and "Smooth_Score" to assess the efficacy of each configuration in our ablation study as shown in Table.7. It is observable that all instances of "Train_Loss" are consistently lower than "Validation_loss", indicating that the model is not overfitting seriously. An interesting observation is that the combined effects of two modules do not always surpass those of models using only a single module. This may be attributed to the fact that the subsequent module may not effectively exploit the information mined by the first module. Each module possesses unique functionalities, and omitting any module might lead to the underutilization of certain features.

However, it is evident that, when initialized with the same parameters, the model encompassing all components (ALL) indeed performs relatively better across all evaluated metrics, except for the "Train_Loss", where it is outperformed by Model C. Nevertheless, model ALL's "Validation_loss" and "Smooth_Score" are superior to Model C's, suggesting that Model ALL's comparative disadvantage is likely due to Model C's overfitting.

V. CONCLUSION

This paper proposes a novel Residual-CNN-GRU-Attention model capable of global data mining. It addresses the issue of steep curve fitting in short-term load forecasting/filling problems. The CNN component of the model extracts temporal information without affecting the spatial relationship among the load consumption of different users. In contrast, the GRU component extracts temporal relationships within the load consumption of the same user for future prediction/filling. Subsequently, the attention mechanism mines the relationships among the data by assigning different weights to different temporal states, helping the model to learn more accurately and obtain more global prediction/filling capability. The model is compared with other deep learning models to demonstrate its superior accuracy and smoothness in residential load forecasting. It achieves a 15.1% increase in prediction accuracy and a 40.8% increase in smoothness compared to the CNN model.

Furthermore, the analysis of algorithm time cost shows that the model can achieve good results with a relatively small increase in computing power consumption, avoiding the issue of high computing power requirements for Transformer. Future work aims to train and adapt the model with load data from different regions and time periods, optimize the model architecture, and make progress in accuracy, time consumption, and smoothness. Ultimately, the goal is to accurately predict/fill load data using data from a specific region and time period.

REFERENCES

[1] C. Genes, I. Esnaola, S. M. Perlaza, L. F. Ochoa, and D. Coca, "Robust recovery of missing data in electricity distribution systems," *IEEE Trans. Smart Grid*, vol. 10, no. 4, pp. 4057–4067, Jul. 2019.

[2] Y. Kim, H.-G. Son, and S. Kim, "Short term electricity load forecasting for institutional buildings," *Energy Rep.*, vol. 5, pp. 1270–1280, Nov. 2019.

[3] H. Takeda, Y. Tamura, and S. Sato, "Using the ensemble Kalman filter for electricity load forecasting and analysis," *Energy*, vol. 104, pp. 184–198, Jun. 2016.

[4] C. Roach, R. Hyndman, and S. Ben Taieb, "Non-linear mixed-effects models for time series forecasting of smart meter demand," *J. Forecasting*, vol. 40, no. 6, pp. 1118–1130, Sep. 2021.

[5] Y. Goude, R. Nedellec, and N. Kong, "Local short and middle term electricity load forecasting with semi-parametric additive models," *IEEE Trans. Smart Grid*, vol. 5, no. 1, pp. 440–446, Jan. 2014.

[6] M. Barman, N. B. Dev Choudhury, and S. Sutradhar, "A regional hybrid GOA-SVM model based on similar day approach for short-term load forecasting in Assam, India," *Energy*, vol. 145, pp. 710–720, Feb. 2018.

[7] H. H. H. Aly, "A proposed intelligent short-term load forecasting hybrid models of ANN, WNN and KF based on clustering techniques for smart grid," *Electr. Power Syst. Res.*, vol. 182, May 2020, Art. no. 106191.

[8] G.-F. Fan, Y.-H. Guo, J.-M. Zheng, and W.-C. Hong, "Application of the weighted K-Nearest neighbor algorithm for short-term load forecasting," *Energies*, vol. 12, no. 5, p. 916, Mar. 2019.

[9] B. Goehry, Y. Goude, P. Massart, and J.-M. Poggi, "Aggregation of multi-scale experts for bottom-up load forecasting," *IEEE Trans. Smart Grid*, vol. 11, no. 3, pp. 1895–1904, May 2020.

[10] S. Liu, M. Zhang, P. Kadam, and C. J. Kuo, *3D Point Cloud Analysis: Traditional, Deep Learning, and Explainable Machine Learning Methods*. Cham, Switzerland: Springer, 2021.

[11] S. Bahrami, R.-A. Hooshmand, and M. Parastegari, "Short term electric load forecasting by wavelet transform and grey model improved by PSO (particle swarm optimization) algorithm," *Energy*, vol. 72, pp. 434–442, Aug. 2014.

[12] E. Mocanu, P. H. Nguyen, M. Gibescu, and W. L. Kling, "Deep learning for estimating building energy consumption," *Sustain. Energy, Grids Netw.*, vol. 6, pp. 91–99, Jun. 2016.

[13] H. W. Loh, C. P. Ooi, J. Vicnesh, S. L. Oh, O. Faust, A. Gertych, and U. R. Acharya, "Automated detection of sleep stages using deep learning techniques: A systematic review of the last decade (2010–2020)," *Appl. Sci.*, vol. 10, no. 24, p. 8963, Dec. 2020.

[14] M. N. Fekri, H. Patel, K. Grolinger, and V. Sharma, "Deep learning for load forecasting with smart meter data: Online adaptive recurrent neural network," *Appl. Energy*, vol. 282, Jan. 2021, Art. no. 116177.

[15] H. Zang, R. Xu, L. Cheng, T. Ding, L. Liu, Z. Wei, and G. Sun, "Residential load forecasting based on LSTM fusing self-attention mechanism with pooling," *Energy*, vol. 229, Aug. 2021, Art. no. 120682.

[16] S. F. Stefenon, L. O. Seman, L. S. Aquino, and L. D. S. Coelho, "Wavelet-Seq2Seq-LSTM with attention for time series forecasting of level of dams in hydroelectric power plants," *Energy*, vol. 274, Jul. 2023, Art. no. 127350.

[17] S. Jung, J. Moon, S. Park, and E. Hwang, "An attention-based multilayer GRU model for multistep-ahead short-term load forecasting," *Sensors*, vol. 21, no. 5, p. 1639, Feb. 2021.

[18] Y. Han, G. Huang, S. Song, L. Yang, H. Wang, and Y. Wang, "Dynamic neural networks: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 11, pp. 7436–7456, Nov. 2022.

[19] L. Maraaba, M. Almuhaini, M. Habli, and M. Khalid, "Neural networks based dynamic load modeling for power system reliability assessment," *Sustainability*, vol. 15, no. 6, p. 5403, Mar. 2023.

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017, *arXiv:1706.03762*.

[21] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, Dec. 2014, pp. 3104–3112.

[22] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," 2014, *arXiv:1406.1078*.

[23] C. Olah. (2015). *Understanding LSTM Networks*. Accessed: Apr. 17, 2024. [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[24] S. D. Siregar, A. Rindang, and P. C. Ayu, "Principle component analysis (PCA)-classification of arabica green bean coffee of North Sumatera using FT–NIRS," *IOP Conf. Series. Earth Environ. Sci.*, vol. 454, no. 1, 2020, Art. no. 12046.

[25] (2017). *UCI Machine Learning Repository*. [Online]. Available: https://archive.ics.uci.edu/

[26] D. Picard, "Torch.manual_seed(3407) is all you need: On the influence of random seeds in deep learning architectures for computer vision," 2021, *arXiv:2109.08203*.

WUTAO XIONG received the B.S. degree in communication engineering from the University of Electronic Science and Technology of China and the University of Glasgow, in 2024. He is currently pursuing the M.S. degree with Sichuan University. His research interests include deep learning model construction and application, focusing on natural language processing, data analysis, and large model applications.