IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 27 May 2023, accepted 19 June 2023, date of publication 22 June 2023, date of current version 18 March 2024.

*Digital Object Identifier 10.1109/ACCESS.2023.3288817*

RESEARCH ARTICLE

# Deep Learning Model-Based Demand Forecasting for Secondary Water Supply in Residential Communities: A Case Study of Shanghai City, China

**DALI LI [1] AND QINGWEN FU [2]**
[1] Division of Asset and Laboratory Management, Shanghai Normal University, Shanghai 200234, China
[2] College of Information, Mechanical and Electrical Engineering, Shanghai Normal University, Shanghai 200234, China

Corresponding author: Dali Li (darly_lee@163.com)

**ABSTRACT** To promote intelligent water services and accelerate the water industry's modernization process, accurately predicting regional residents' water demand and reducing energy consumption for secondary water supply is a major challenge for scientific scheduling and efficient management of urban water supply. This paper proposes a deep learning-based approach for demand forecasting in residential communities. The approach first identifies and corrects outliers in raw water supply data, and incorporates additional features such as epidemics and meteorological information. A long and short time Transformer model (LTMFormer) is then proposed, combining the recursive mechanism of the LSTM model and the parallel mechanism of the Transformer to achieve parallel output in the long-time series modeling task, improving both the prediction length and accuracy of the model. We evaluate our model on a metering dataset of 20 cells in Shanghai and compare it to traditional deep-learning models. Our experimental results demonstrate that the proposed model outperforms other deep learning models, achieving MSE, RMSE, and MAE scores of 3.337, 4.536, and 1.848 respectively on the test set. These results provide a theoretical and technical basis for further safeguarding public water health and meeting the growing demand for better urban water management.

**INDEX TERMS** Water demand prediction, outlier identification, LSTM model, transformer model, deep learning.

## I. INTRODUCTION

In recent years, with the rapid development of urbanization and continuous population growth, the construction of residential communities has increasingly become a primary focus of urban planning [1], [2]. Alongside this, the fast-paced development of urban water supply systems has made secondary water supply technology a vital means to enhance the quality of tap water and improve the living standards of residents. Throughout this process, accurately forecasting secondary water supply demand in residential communities has become an essential issue. Currently, traditional prediction methods rely on experience and statistical models, which possess certain limitations and shortcomings. However, with the constant progress and integration of deep learning technology in recent years, predicting secondary water supply demand in residential communities based on deep learning models have been introduced into the research field. By training deep learning algorithms on historical data, future demand for secondary water supply can be more precisely predicted, leading to better community water supply efficiency and optimized water supply plans. Therefore, this paper aims to explore the prediction method and application prospects of secondary water supply demand in residential communities based on deep learning models. Against this backdrop, attention towards the rational utilization and

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

protection of water resources has intensified, and the development and application of secondary water supply technology have garnered increasing attention from governments and citizens alike.

Water demand forecasting is a topic of widespread research globally. In recent times, machine learning algorithms have gained popularity in this area. For instance, Hipni, Liu, and Msiza [3], [4] employed support vector machines (SVM) for water demand forecasting. Similarly, ARIMA models have been utilized for urban water demand prediction [5], [6], [7], [8]. Random Forest (RF) was used by Kumar et al. [9] to solve the problem. Artificial neural networks have also been refined to handle the nonlinear feature-fitting issue in water demand forecasting. Although machine learning-based models have exhibited greater progress than statistical-based methods in water demand forecasting, the shallow structure of classical machine learning models may not be as effective in predicting water demand series with complex data and long histories.

Recently, rapid advances in deep learning have led to the emergence of various deep learning-based models that have revolutionized water demand prediction. For example, Xu et al. [10] employed a continuous deep belief neural network (CDBNN) model based on chaos theory for urban water demand prediction, which outperformed feedforward neural networks, support vector regression, and generalized regression neural networks. Meanwhile, Guo [11] developed a novel hybrid model for urban water demand prediction using temporal convolutional neural networks (TCN), discrete wavelet transform (DWT), and random forest (RF). The experimental outcomes suggest that this model is highly effective in predicting urban water demand.

With the continuous development of technology, Transformer [12] has been gradually applied in various fields. Initially utilized in natural language processing for machine translation tasks, the Transformer model has evolved to encompass BERT [13] language models, as well as the Vision Transformer [14], which has found applications in computer vision. The Transformer model's powerful feature extraction capability has been demonstrated across all these areas. However, in the time series prediction field, all compared baselines use autoregressive prediction, which inevitably generates error accumulation effects. The multi-headed self-attentive mechanism is the primary driving force behind the Transformer architecture, endowed with remarkable abilities to extract semantic correlations between pairs of elements in long sequences (e.g., words in text or 2D patches in images). This process is alignment invariant, meaning that it is not affected by order changes. However, in time series analysis, we are mainly concerned with modeling the temporal dynamics between a set of consecutive points, where the order itself often exerts significant influence.

When it comes to long-term feature extraction, LSTMs [15] represent the input sequence they learn as a state vector before moving on to future tokens. Although LSTMs solve the gradient vanishing problem, they are still prone to gradient explosion and cannot parallelize output. On the other hand, the Transformer has higher bandwidth. For example, in the Encoder-Decoder Transformer model, the Decoder can directly process each token in the input sequence, including those already decoded. However, since the complexity of the attention mechanism grows exponentially with the sequence length, it is not well-suited for handling long-term tasks.

Bao et al. [16] applied a hybrid LSTM and autoencoder model to process financial time series data and demonstrated the good performance of this hybrid model. Andayani et al. [17] used a hybrid LSTM and Transformer model to learn long-term dependencies in speech signals and perform sentiment classification. Delgado-Santos et al. [18] proposed a hybrid LSTM and Transformer model for temporal modeling and natural language prediction in behavioral biometrics. These studies indicate that the hybrid LSTM and Transformer model is a versatile and effective approach that can be used for sequential data modeling across different domains, particularly in scenarios involving time series data [19], [20].

Building upon the theoretical foundations and relevant issues discussed above, this paper proposes a Long-Term Memory Transformer (LTMFormer) which leverages the unique recursive structure of LSTM to enable long-term sequence prediction for water supply demand while being read in parallel. Specifically, the contributions of this paper are as follows.

1) We effectively combine LSTM neural network and Transformer architecture for the first time and use them in the field of urban water supply prediction. Our proposed approach leverages the strengths of both models to address the challenges of predicting water demand in urban areas, particularly in fast-growing cities like Shanghai. By doing so, we offer a novel solution that improves the accuracy and robustness of existing deep learning-based methods.
2) We have conducted case experiments with the actual smart meter dataset in some neighborhoods in Shanghai, and the experimental results show that the proposed model is ahead of traditional deep learning models in all indexes. The empirical evaluation demonstrates the effectiveness and practicality of our approach in real-world settings. By comparing the performance of our proposed model with several state-of-the-art baselines, we show that it achieves superior performance in terms of accuracy, efficiency, and scalability. These findings suggest that our proposed model has significant potential for wider adoption in the field of urban water resource management.
3) This paper organically combines deep learning technology and urban water consumption prediction problem, providing a new solution idea for this field. By integrating deep learning techniques with the problem of predicting water demand in urban areas, we offer a

fresh perspective on the challenges and opportunities of using artificial intelligence to manage water resources more effectively. Our research contributes to the growing body of literature on data-driven approaches to urban water management, paving the way for further research and development in this field. In summary, our study offers a valuable contribution to the academic community by showcasing the potential of deep learning for addressing complex environmental challenges.

## II. DATA AND DATA PROCESSING

### A. DATA INTRODUCTION

To predict water demand, we utilized a deep learning model that incorporated external factors such as weather data, COVID-19 case numbers, and sensor data from major intersections. The specific sources of these data were the China Meteorological Data Sharing Service for weather data, the Shanghai Municipal Health Commission website for COVID-19 case numbers, and sensors installed at major intersections for daily interval flows and hourly interval flows. Additionally, we obtained water supply data from actual water consumption data desensitized in a residential district in Shanghai, as well as historical readings of every 5-minute interval, hourly flow rate readings, and daily flow rate readings of the flow meters installed behind the secondary water supply pumps in the same residential district. Finally, we obtained total meter readings of the smart water meters in the residential district through a big data platform. The specific data are described as follows:

#### 1) DAILY INTERVAL TRAFFIC DATASET

The daily interval traffic dataset in this paper provides a visual presentation of the data, as shown in Figure 1. This dataset includes cumulative water flow from a specific time point to the previous time point, which is referred to as interval flow. However, due to infrequent meter readings, anomalous values with large intervals were observed in the training set but not in the test set. The dataset consists of 6 cells (01-06) with consistent coding among several tables. The interval flows for cells 01 to 06 are represented by flow_1, flow_2, ..., and flow_6, respectively. This dataset provides daily interval flows as described below. Note that some folds in the graphs may be interrupted at certain points, so default values may need to be processed subsequently. As shown in Figure 1, the line graph displays the daily interval traffic dataset.

#### 2) HOURLY INTERVAL TRAFFIC DATASET

The hourly interval traffic dataset provided in this paper is presented in Figure 2. For observation purposes, we visualized the hourly interval volume data for four of the cells, as shown in Figure 3. This dataset contains outliers such as missing values, as well as very large and very small values.

#### 3) EVERY 5-MINUTE INTERVAL TRAFFIC DATASET

This dataset provides the interval traffic per 5 minutes, which is shown in Figure 4, and we visualize it.

#### 4) SHANGHAI WEATHER DATA

This data provides information on the weather conditions in Shanghai and can be utilized as input features for enhancing features. It includes various parameters such as precipitation (R) measured in 0.1 mm, wind direction (fx) measured in degrees, temperature (T) measured in degrees Celsius, relative humidity (U), wind speed (fs) measured in 0.1 m/s, visibility (V) measured in meters, and barometric pressure (P) measured in hectopascals. These parameters are depicted in Figure 5.

#### 5) SHANGHAI EPIDEMIC DATA

This dataset provides epidemic data for Shanghai and includes parameters such as the number of severe cases (zz), the number of critical cases (wz), the number of confirmed cases (xzqz), the number of new discharges (xzcy), the number of new deaths (xzsw), the number of people currently receiving isolation treatment (glzl), and the number of people under medical observation (yxgc). These parameters are depicted in Figure 6.

### B. DATA PROCESSING METHODS

#### 1) DATA PRE-PROCESSING

Based on the above graphs, it can be observed that the data contains abnormal values such as NaN (Not a Number) values, negative values, abnormally large values, and so on. Therefore, we performed the following data processing steps to clean up the data:

- To address the issues highlighted in the above findings, we adopted the following data preprocessing strategy: negative and abnormal values were set as null values, and then the original null values together with those newly set were filled.
- The null values were identified by simulating the normal distribution of the data. Specifically, values greater than 90% of the data were considered as high, while those less than 90% were regarded as low. The threshold values were calculated based on the high and low values, and all values exceeding the thresholds were marked as null. This approach helps to eliminate great outliers observed in the original data, leading to much smaller standard deviation. However, it also resulted in a considerable increase in the number of null values.
- For filling the null values, we employed a strategy that preserves the mean and variance of the original data as closely as possible. We used the data from the same moment of the previous/next day to fill the missing values based on our understanding of the data composition. In cases where there was no available data for the previous/next day, we fixed the null values to zero.

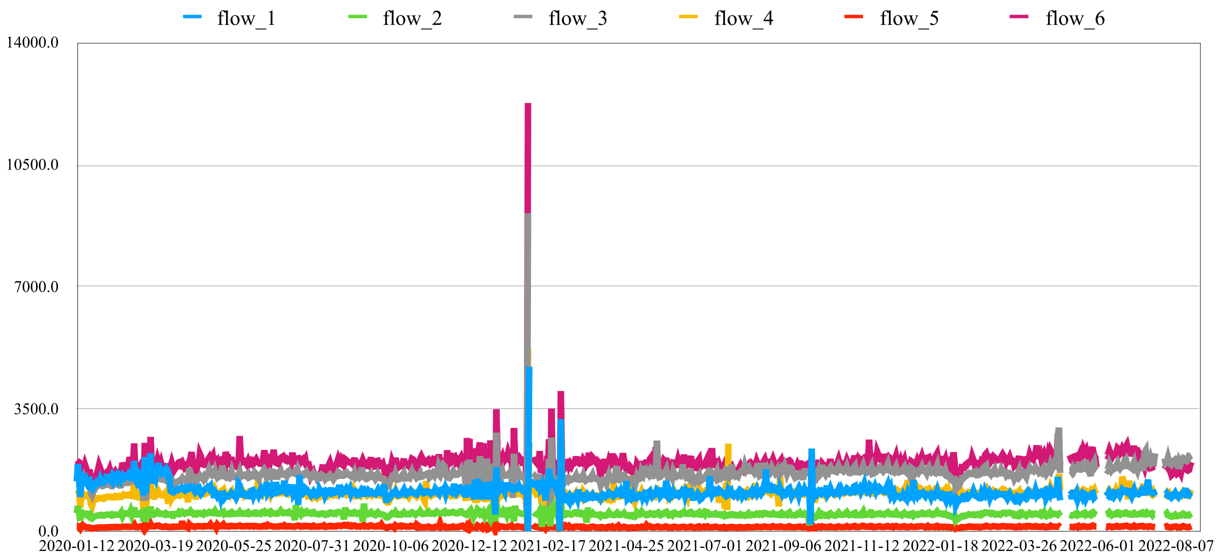


**FIGURE 1.** Line graph of daily interval traffic data set.

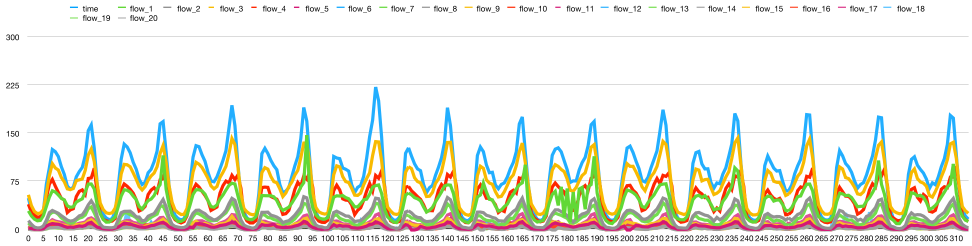


**FIGURE 2.** Line graph of hourly interval flow data set.

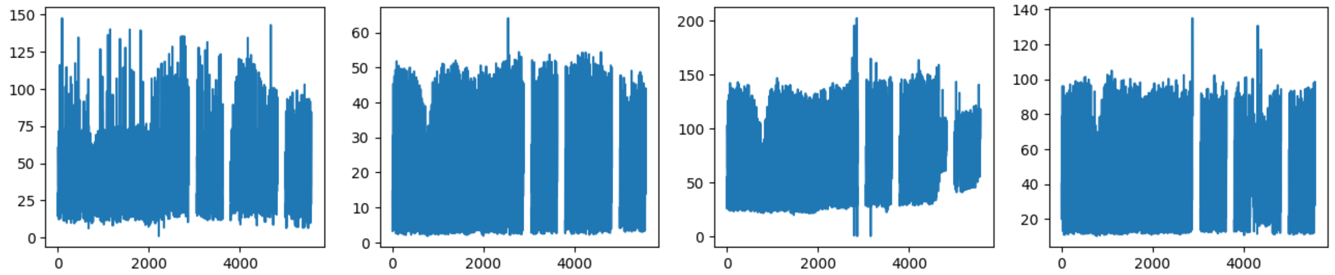


**FIGURE 3.** Visual display of four plots.

### 2) TRAINING SET, TEST SET CONSTRUCTION

In this study, we constructed the time series training set based on the process outlined in Figure 7. We adopted a segmented training approach that followed the following basic rules:

- Training set 1 can be used to predict test set 1.
- Training sets 1, 2, and 3 can be used to predict test set 3.
- Semi-supervised learning can be used to predict test set 2 using training set 1, test set 1, and training set 2.
- It is forbidden to use training set 3 to predict test sets 1 and 2.

In terms of analyzing the specific data, the training set part contains a total of 5736 hours of recorded data, and the test set part has a total of 672 hours of recorded data, which are divided into four segments, each segment is 168 data for 7 days, and its various time segments are [2022-05-01 01:00:00 ∼ 2022-05-08 00:00:00], [2022-06-01 01:00:00 ∼ 2022-06-08 00:00:00], [2022-07 -21 01:00:00 ∼ 2022-07-28 00:00:00], [2022-08-21 01:00:00 ∼ 2022-08-28 00:00:00]. The original data input features are 20, i.e., representing flow_1 to flow_20, which are the features needed to be predicted in this paper.

Here in this paper, the length of the test set T=168 is the time span of each constructed data, then, the data of each T length is X, and the data of the immediately

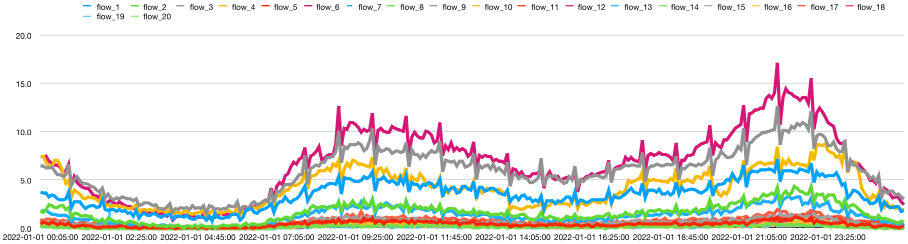


**FIGURE 4.** Line graph of the flow data set for each five-minute interval.

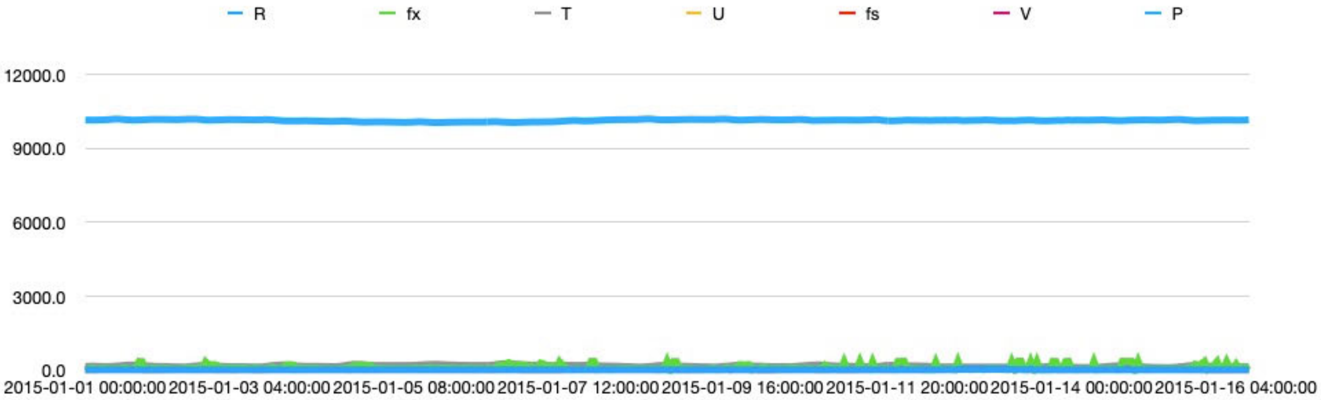


**FIGURE 5.** Shanghai weather data visualization.

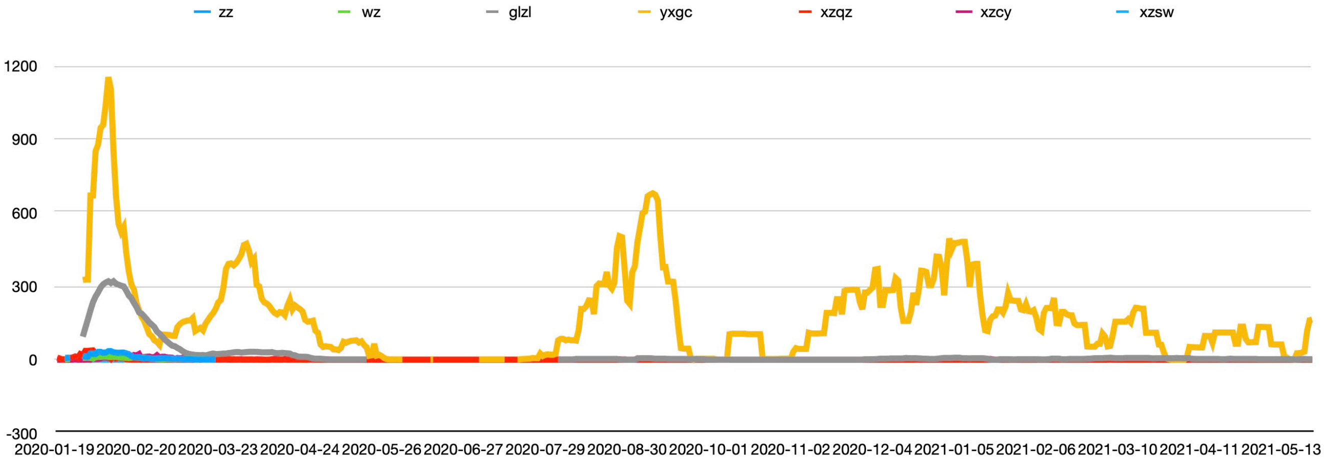


**FIGURE 6.** Visualization of the epidemic data in shanghai.

adjacent length T is Y, which is generated with a sliding window w=1 hour hour rolling. Each time point t of X includes m feature values, such as flow_1, flow_2, etc., and constructed features such as weather, epidemic, etc. After constructing the data, the corresponding training data are extracted from the sequence according to the time points of the four test sets. The final test data, in fact, is only 4, that is, the corresponding 4 test time points before T of the data.

## III. RESEARCH DESIGN

We first introduce the research motivation of this paper, mainly explaining why LSTM and Transformer should be used for water demand prediction problems. This is also an important driving force for our research. Secondly, we describe the details of the model in detail, reviewing the conventional LSTM model and Transformer model, and then carefully describing how to effectively combine them for use in water demand prediction problems.

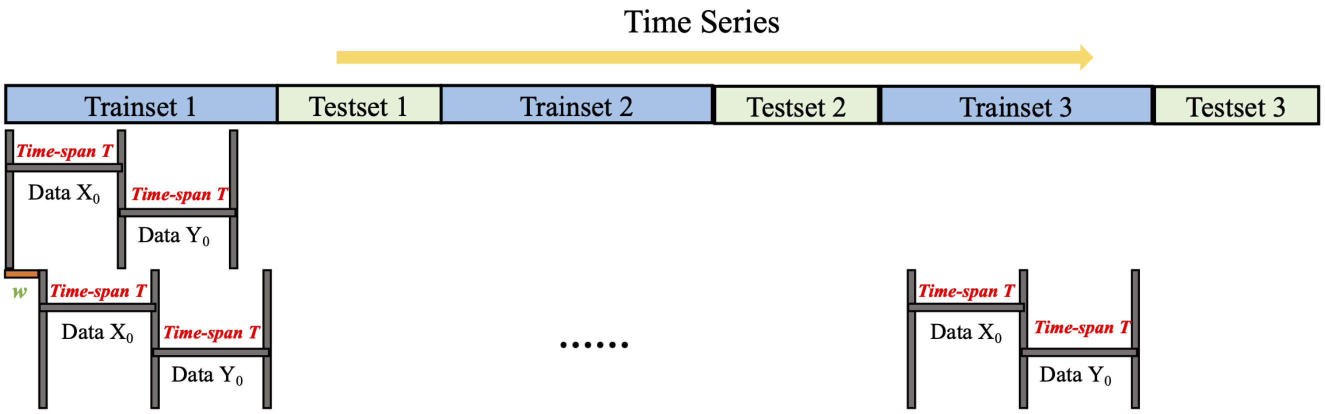


**FIGURE 7.** Flow chart of dataset construction.

### A. RESEARCH MOTIVATION

To maximize the accuracy of demand forecasting in practical applications, deep learning models have been widely used in the water supply field. However, existing research typically only adopts a single architecture, such as LSTM or Transformer, and does not fully utilize the advantages of these models.

LSTM, as a type of recurrent neural network (RNN), excels at processing sequence data by introducing memory cells, input and forget gates, and output gates. It can accurately predict future time periods by capturing temporal patterns and dependencies in historical data. In contrast, Transformer is a relatively new model architecture whose self-attention mechanism makes it particularly adept at handling long sequences of time series data where maintaining long-term dependencies is crucial for accurate forecasting.

Therefore, this study chooses to mix LSTM and Transformer to fully leverage their strengths in processing sequence data, thereby improving the accuracy and efficiency of demand forecasting. Additionally, we focus on secondary water supply in residential communities in Shanghai, China, which is a challenging research field due to the existence of many complex variables and non-linear relationships that are difficult to capture. By using hybrid models, we hope to improve the accuracy of our predictions and provide guidance and reference for other research in this field.

In summary, the motivation of this study is to explore and improve demand forecasting methods based on deep learning, and apply them to the secondary water supply system in residential communities in Shanghai. We believe that this work can greatly improve prediction accuracy and provide support for more informed decision-making and resource management.

### B. MODEL DETAILS

LTMFormer is a deep learning model that combines the advantages of both Transformer and LSTM architectures to be able to make predictions on time series data.

In this model, the input data is first processed using the Transformer to output a set of feature vectors in parallel. These feature vectors contain global information about the original data and can be computed quickly and efficiently due to Transformer's parallel computing capabilities. These feature vectors are then fed into the LSTM to extract relevant information about the time series. the memory unit and gating mechanism of the LSTM allow the model to efficiently capture the evolution pattern of the time series and generate the final prediction results.

Therefore, the LTMFormer model is able to use the Transformer to process the input data to extract global features, and also to fully utilize the time series modeling capability of the LSTM to achieve more accurate and efficient water supply demand time series forecasting. The model structure is shown in Figure 8.

#### 1) INPUT LAYER

When inputting time series data into the model, we need to determine the features for each time node. In this paper, we use a variety of features including epidemiological data, meteorological data, and dates, and represent each time node as a vector containing 25 features. Since we set the length of the test set to 168 time nodes and used a 20-cell LSTM model for parallel prediction, we input each cell as a feature.

Specifically, each time node contains 25 features, so we represent the time series data consisting of all time nodes as a matrix with the dimension of [25 × 168]. In this input matrix, each column corresponds to the feature vector of the current time node, 25 in total, and each row corresponds to a different time node.

Such an input layer is designed to integrate the rich feature information into the model effectively and to improve the training efficiency and prediction accuracy by using the parallel computing capability of the Transformer model.

#### 2) FULL CONNECTED DENSE LAYER

In the LTMFormer model, the fully connected layer is utilized to map the input data to the hidden layer feature space and

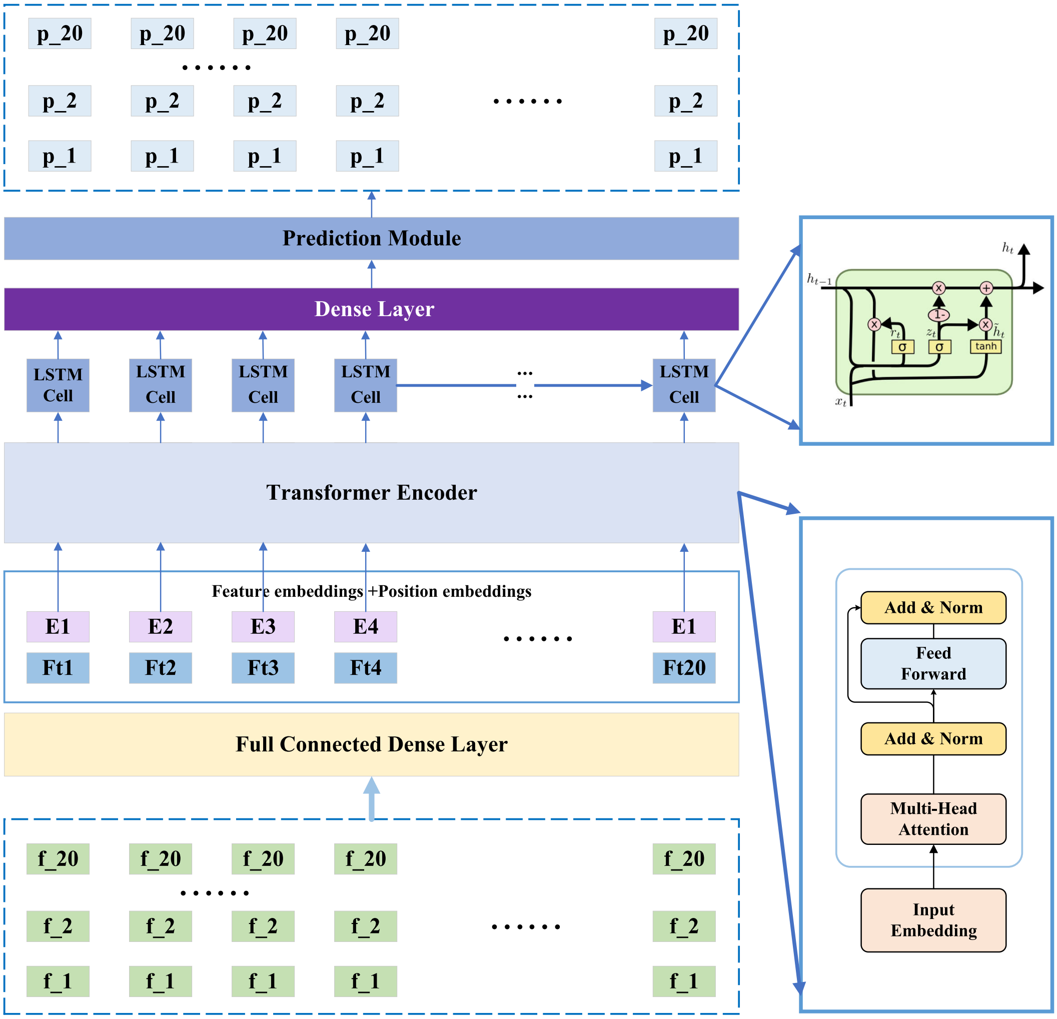


**FIGURE 8.** Model architecture diagram.

transfer the learned distributed representation to the subsequent Transformer module. This layer plays a key role in transforming the original time series data into a new vector representation that captures higher-level features.

Specifically, we first represent the time series data as a [25 × 168] matrix, where each element corresponds to the feature value of a time node. Then, we flatten this matrix into a vector of length 4200 and pass it through a linear transformation by a fully connected layer. The output of this layer is the hidden layer feature vector which has a dimensionality controlled by the model hyperparameters.

Next, we feed the hidden layer feature vector to the Transformer module for parallel computation. In this process, the features of each time node are considered as an input vector, and intensive self-attentive computations and feedforward neural network computations are performed against these vectors to obtain the predicted output for that time node.

Overall, the fully connected layer serves as a critical component in the LTMFormer model by converting the input data into a new hidden layer feature space that captures higher-level information. This design allows the model to learn more complex temporal patterns and achieve better prediction performance.

### 3) FEATURE EMBEDDINGS + POSITION EMBEDDINGS

When using Transformer Encoder to process sequential data, we need to consider how to incorporate position information of the sequence into the model. Since Transformer Encoder itself cannot recognize the positional relationship of the sequence automatically, we need to solve this problem by adding Position Embeddings.

In the LTMFormer model, we adopt a special way to implement Position Embeddings, which is cascading with Feature Embeddings. Specifically, after the feature vector output from the fully connected layer, we add a vector with the same length as the input sequence, where each element corresponds to the position of that time node in the sequence. This vector is called Position Embeddings, which provides information about the order of the time series to the model.

Then, we cascade Feature Embeddings and Position Embeddings to obtain a new cascaded vector, which serves as the input of the Transformer Encoder module. In this module, Position Embeddings and Feature Embeddings participate in computation together, enabling the model to better understand the temporal characteristics of the sequential data and generate corresponding prediction results.

In summary, the LTMFormer model utilizes Feature Embeddings and Position Embeddings to form a cascaded vector, effectively transforming the original time series data into a format that can be input to the Transformer Encoder module. This design enables the model to predict future secondary water supply demand more accurately.

### 4) TRANSFORMER ENCODER

Each layer of the Transformer Encoder structure contains two sub-layers, a multi-headed attention layer and a feed-forward connection layer. Each sublayer is followed by a residual connection and a normalization layer.

Its computational process.

1. The input is passed through an input embedding layer, which means that the sound signal or the text is transformed into a vector form.
2. The location code is added, these signals are without location information, here the lo-cation information is added.
   Since the Transformer model does not have the iterative operation of RNN, it is necessary to provide the location information, and here the linear transformation of Sine and Cosine functions is used to provide the location information to the model.

$$\mathrm{PE}(\mathrm{pos}, 2\mathrm{i}) = \sin\left(\mathrm{pos}/10000^{2\mathrm{i}/\mathrm{dmodel}}\right)$$
$$\mathrm{PE}(\mathrm{pos}, 2\mathrm{i}+1) = \cos\left(\mathrm{pos}/10000^{2\mathrm{i}/\mathrm{dmodel}}\right) \tag{1}$$

3. Multi-head attention layer. Multi-head attention is to split the query, key and value parameters multiple times while the overall number of parameters remains unchanged, and each set of split parameters is mapped to different subspaces of the high-dimensional space to calculate the attention weights so as to focus on different parts of the input. After several parallel calculations, the attention information in all subspaces is finally combined. Its structure is shown in Figure 9.

$$MultiHead(Q, K, V) = Concat(head1, \ldots, headh)W_0 \tag{2}$$

$$head\_i = Attention\left(\mathrm{QW}_{\mathrm{i}}^{\mathrm{Q}}, \mathrm{KW}_{\mathrm{i}}^{\mathrm{K}}, \mathrm{VW}W_{\mathrm{i}}^{\mathrm{V}}\right) \tag{3}$$

4. The output of the previous step is added to a residual connection to preserve information from previous layers and avoid vanishing gradients. Then, it is passed through a layer normalization process to normalize the values across features for each individual sample in the batch. This step helps to stabilize the learning process and improve the model's generalization ability.
5. The output of the previous step is passed through a Multi-Layer Perceptron (MLP) layer, which is a fully connected feed-forward network with nonlinear activation functions. The purpose of this layer is to learn higher-level representations based on the extracted features from the multi-head attention layer. Specifically, the MLP layer applies a linear transformation followed by a non-linear activation function to each feature separately.
6. The output of the MLP layer is added to a residual connection and then passed through another layer normalization process. This step helps to manage the scale of the output and stabilize the training process.

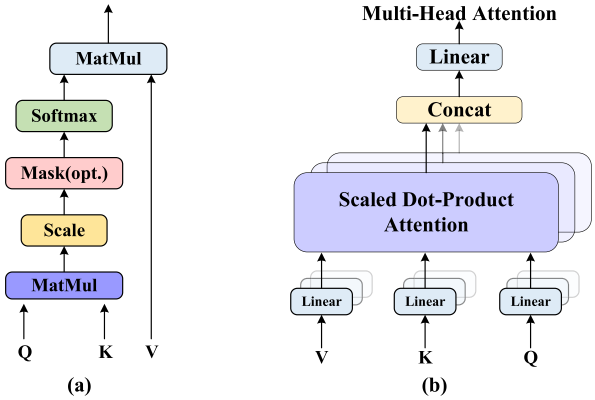


**FIGURE 9.** Attention mechanism diagram; (a) scaled dot product attention(b) multi-head attention.

### 5) LSTM CELL

In many sequence modeling tasks, the Long Short-Term Memory (LSTM) architecture has been proven to be effective due to its robustness and stability in modeling long-range dependencies. The main innovation of the LSTM is its storage unit, denoted as ct, which accumulates state information. This unit is accessed, written, and cleared by several self-parameterizing control gates. Each time a new input is received, if the input gate is activated, the information is accumulated in the cell. Additionally, if the forget gate ft is opened, the previous cell state ct-1 is "forgotten". Whether the newest cell output ct is propagated to the final state ht is further controlled by the output gate ot. Utilizing storage cells and gates to control the flow of information helps prevent the gradients from disappearing too quickly, which can be problematic for vanilla RNN models.

The Fully-Connected LSTM (FC-LSTM) can be seen as a multivariate version of the LSTM, where the inputs, cell outputs, and states are all one-dimensional vectors. In this paper, we introduce our original FC-LSTM formulation, with key equations shown below:

$$\begin{aligned} i_t &= \sigma\left(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci} \circ c_{t-1} + b_i\right) \\ f_t &= \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf} \circ c_{t-1} + b_f\right) \\ c_t &= f_t \circ c_{t-1} + i_t \circ \tanh\left(W_{xc}x_t + W_{hc}h_{t-1} + b_c\right) \\ o_t &= \sigma\left(W_{xo}x_t + W_{ho}h_{t-1} + W_{co} \circ c_t + b_o\right) \\ h_t &= o_t \circ \tanh\left(c_t\right) \end{aligned} \tag{4}$$

The LSTM architecture has been shown to be effective in modeling long-range dependencies and is widely used in various sequence modeling tasks. However, for multivariate time series data, the Fully-Connected LSTM (FC-LSTM) model extends the LSTM architecture to handle multiple correlated time series simultaneously. This makes it suitable for real-world applications that require modeling complex interactions between multiple variables.

Moreover, by stacking multiple LSTM layers or concatenating them in time, more complex structures can be formed to capture even higher-level patterns in the data. The FC-LSTM architecture has been successfully applied to solve many real-life sequence modeling problems, such as speech recognition, language translation, and financial forecasting, demonstrating its effectiveness and versatility in handling diverse types of data.

#### 6) DENSE LAYER

The Dense Layer plays a crucial role in the proposed model by projecting the feature vector of the LSTM output to a lower-dimensional space through linear projection. This is accomplished by applying a matrix multiplication operation, followed by a bias term, to the LSTM output. The primary purpose of this layer is to reduce the dimensionality of the feature vector, thereby improving the model's efficiency and reducing the risk of overfitting.

In this paper, the dimensionality reduction process is achieved using the Dense Layer, which enables the model to extract more relevant and informative features from the data while discarding irrelevant or redundant information. This process not only improves the performance of the model but also reduces its computational complexity, making it more feasible for real-world applications that involve large-scale datasets.

Overall, the Dense Layer is a critical component of the proposed model, responsible for reducing the dimensionality of the LSTM output and improving the efficiency and effectiveness of the model.

#### 7) PREDICTION MODULE

Despite its simplicity, the prediction module is a critical component of the proposed model, as it directly generates the final predictions for the target variable. By projecting the feature vector to the appropriate output dimension, the prediction module enables the model to capture the complex relationships between multiple variables and generate high-quality forecasts with high accuracy.

Moreover, the output dimension of the prediction module is specifically designed based on the problem requirements. In this paper, the dimension of the prediction is [20×168], where 20 represents the number of cells and 168 represents the time length of the predicted sequence (in hours). This reflects the fact that the model predicts the future values of the target variable for each of the 20 cells over a period of 168 hours.

Overall, while the prediction module is relatively simple compared to other components of the model, it plays a crucial role in generating accurate predictions for the problem at hand. Its linear projection operation effectively maps the feature vector of the input sequence to the appropriate output dimension, allowing the model to learn and capture the dynamics of the underlying data and make accurate forecasts for the future.

### C. MODEL PERFORMANCE EVALUATION METRICS

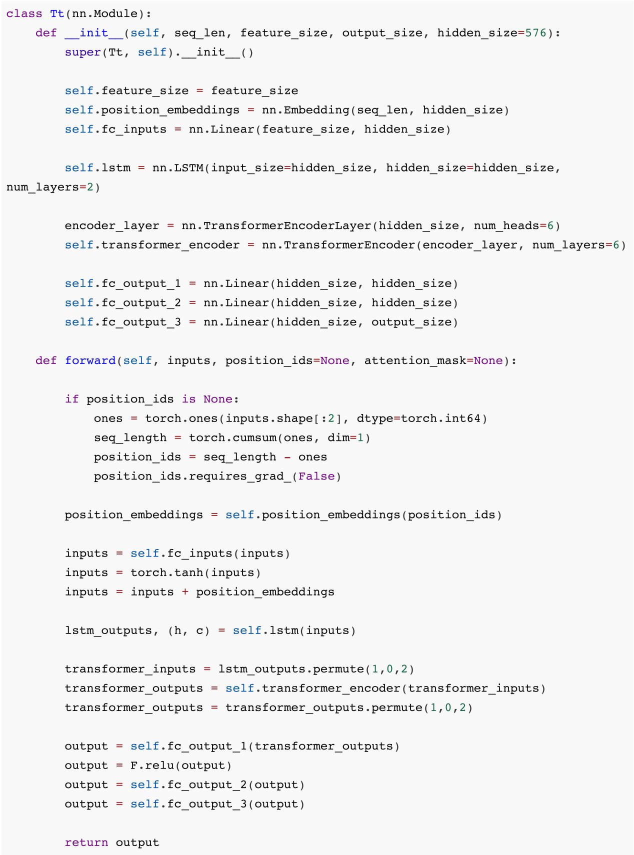
When conducting water supply demand forecasting, deep learning algorithms can be implemented and evaluating the model performance is crucial. Among widely-used metrics, such as MSE and RMSE which prioritize larger errors, MAE is more robust to outliers and extreme values. As a result, it is a better choice when data has high variability or there is a risk of extreme events. Additionally, MAE is easier to interpret than MSE and RMSE because it maintains the target variable's units (e.g. liters per day).

It should be noted that while MSE and RMSE may improve accuracy, they can also increase the model's sensitivity to outliers. Conversely, MAE treats all errors equally, making it less susceptible to being influenced by outliers, but may result in slightly lower accuracy. Ultimately, the choice of error metric depends on the specific characteristics of the dataset and modeling approach, and should be thoughtfully selected based on the analysis goals.

When using different forecasting models for quantitative analysis and evaluation, the evaluation metrics' MSE, RMSE, and MAE are used to assess the accuracy and stability of the model forecasts. These metrics are calculated as follows:

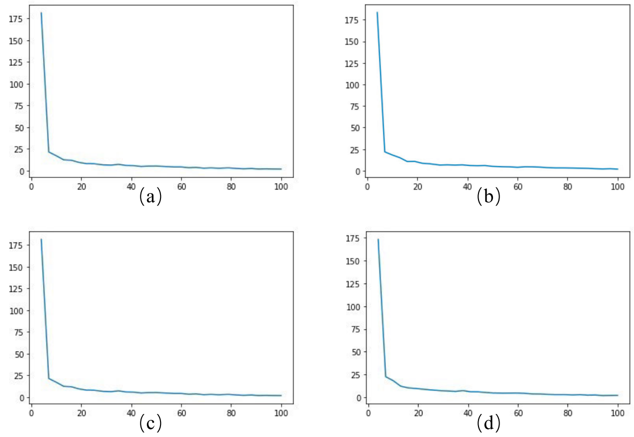
$$\begin{aligned} \text{MSE} &= \frac{1}{m}\sum_{i=1}^{m}\left(y_{\text{test}}^{i} - \hat{y}_{\text{test}}^{i}\right)^2 \\ \text{RMSE} &= \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(y_{\text{test}}^{i} - \hat{y}_{\text{test}}^{i}\right)^2} \\ \text{MAE} &= \frac{1}{m}\sum_{i=1}^{m}\left|\left(y_{\text{test}}^{i} - \hat{y}_{\text{test}}^{i}\right)\right| \end{aligned} \tag{5}$$

In the context of evaluating time series prediction models, MSE, RMSE, and MAE are commonly-used metrics. Here, $y_{\text{test}}^{i}$ and $y_{\text{test}}^{i}$ represent the true and predicted values of a single series, respectively. MSE is a measure of the sum of the squares of the deviations between the predicted and true values. In comparison, RMSE has the same dimension as the original data and reflects the deviation between the true



```
class Tt(nn.Module):
    def __init__(self, seq_len, feature_size, output_size, hidden_size=576):
        super(Tt, self).__init__()

        self.feature_size = feature_size
        self.position_embeddings = nn.Embedding(seq_len, hidden_size)
        self.fc_inputs = nn.Linear(feature_size, hidden_size)

        self.lstm = nn.LSTM(input_size=hidden_size, hidden_size=hidden_size,
num_layers=2)

        encoder_layer = nn.TransformerEncoderLayer(hidden_size, num_heads=6)
        self.transformer_encoder = nn.TransformerEncoder(encoder_layer, num_layers=6)

        self.fc_output_1 = nn.Linear(hidden_size, hidden_size)
        self.fc_output_2 = nn.Linear(hidden_size, hidden_size)
        self.fc_output_3 = nn.Linear(hidden_size, output_size)

    def forward(self, inputs, position_ids=None, attention_mask=None):

        if position_ids is None:
            ones = torch.ones(inputs.shape[:2], dtype=torch.int64)
            seq_length = torch.cumsum(ones, dim=1)
            position_ids = seq_length - ones
            position_ids.requires_grad_(False)

        position_embeddings = self.position_embeddings(position_ids)

        inputs = self.fc_inputs(inputs)
        inputs = torch.tanh(inputs)
        inputs = inputs + position_embeddings

        lstm_outputs, (h, c) = self.lstm(inputs)

        transformer_inputs = lstm_outputs.permute(1,0,2)
        transformer_outputs = self.transformer_encoder(transformer_inputs)
        transformer_outputs = transformer_outputs.permute(1,0,2)

        output = self.fc_output_1(transformer_outputs)
        output = F.relu(output)
        output = self.fc_output_2(output)
        output = self.fc_output_3(output)

        return output
```

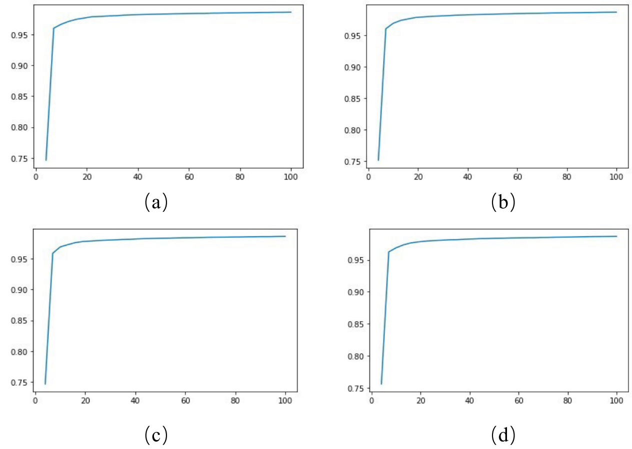
**FIGURE 10.** LTMFormer model building pseudo-code.



(a) (b) (c) (d)

**FIGURE 11.** The loss iteration diagram of the model, where a, b, c, and d denote the loss iteration diagrams of the four stages, subsequently used to predict each of the four test sets, each with a length of 168h.

and predicted values by representing the square root of the ratio of the deviation between the predicted and true values to the number of forecasting tests. On the other hand, MAE represents the mean absolute value of the error, indicating the actual magnitude of the prediction error. The smaller these three evaluation metrics are, the more accurate the model's prediction is considered to be.



(a) (b) (c) (d)

**FIGURE 12.** The Valid Score iteration graph of the model, where a, b, c, and d denote the four stages of the Valid Score iteration graph, and subsequently used to predict each of the four test sets, each with a length of 168h.

**TABLE 1.** Dependency libraries required for the experimental environment.

| Environment names | Version number |
|---|---|
| nvgpu | 0.9.0 |
| regex | 2022.4.24 |
| spacy | 3.3.0 |
| tqdm | 4.62.3 |
| visualdl | 2.2.3 |
| paddlepaddle-gpu | 2.2.2 |
| paddlenlp | 2.3.1 |
| LAC | 2.1.2 |
| shap | 0.40.0 |

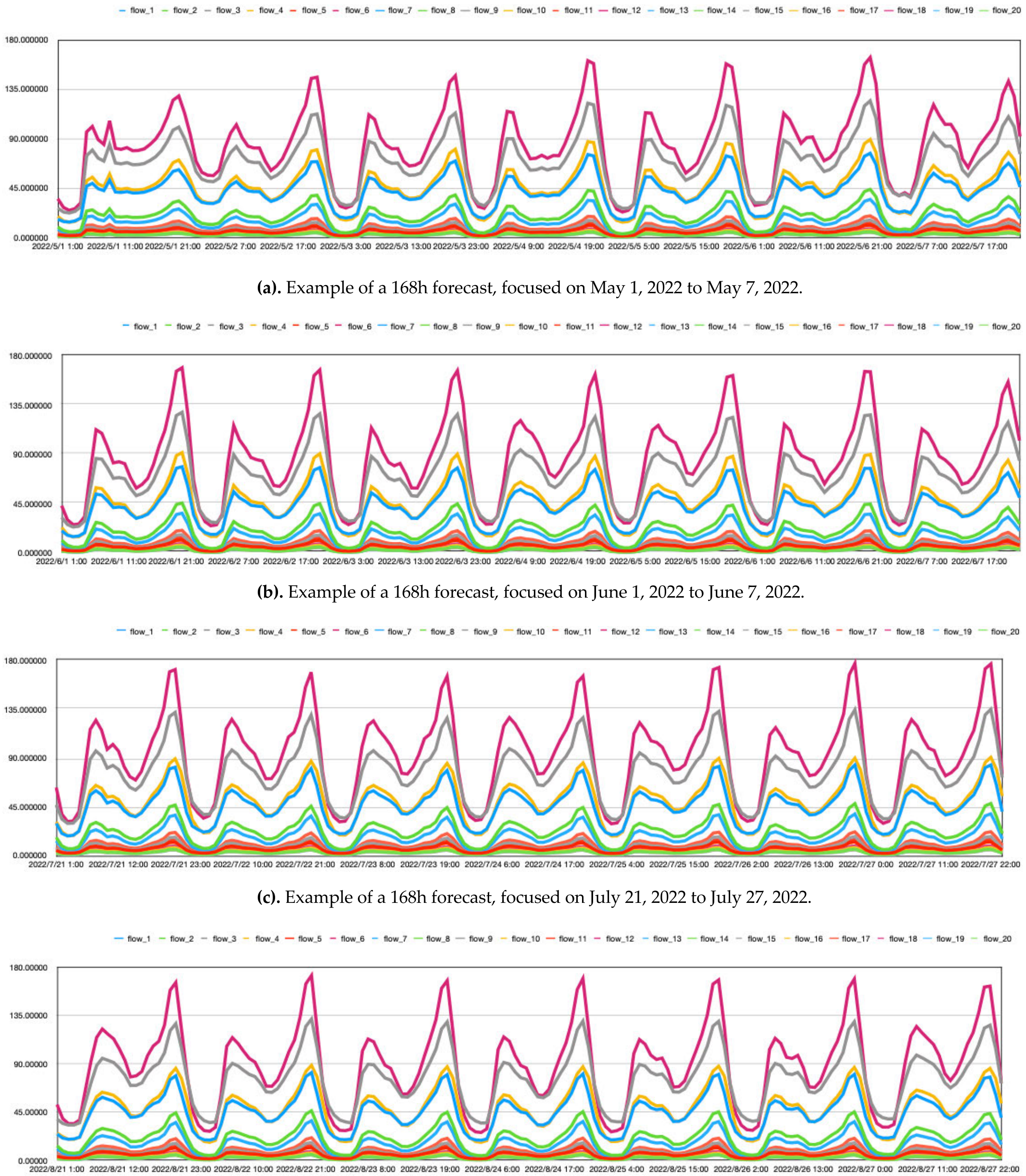
## IV. EXPERIMENTS AND ANALYSIS

### A. EXPERIMENTAL ENVIRONMENT AND HYPERPARAMETER SETTINGS

The experimental environment for the experiments in this paper is a CPU 12-core, where an A100 graphics card with 40GB of video memory size is used, Python environment 3.7.4, and the deep learning framework we use is PaddlePaddle 2.3.2. The dependency libraries required for the experimental environment are specified in Table 1.

The LTMFormer model is a hybrid of LSTM and Transformer used for predicting cell water demand. The model structure includes positional encoding, fully connected layers for input data transformation, LSTM for processing input data, and Transformer Encoder for feature extraction through multiple encoder layers. Finally, two fully connected layers with ReLU activation function are used to transform the output of Transformer Encoder to obtain the final prediction.

The model's hyperparameters include the length of the time series of historical data, the number of features for each time step, the number of predicted values in the output, the size of the hidden layers in LSTM and Transformer Encoder, the number of layers in LSTM, the number of attention heads in Transformer Encoder, the size of the intermediate layer in Transformer Encoder, the dropout probability in the hidden



**(a).** Example of a 168h forecast, focused on May 1, 2022 to May 7, 2022.

**(b).** Example of a 168h forecast, focused on June 1, 2022 to June 7, 2022.

**(c).** Example of a 168h forecast, focused on July 21, 2022 to July 27, 2022.

**(d).** Example of a 168h forecast, focused on August 21, 2022 to August 27, 2022.

**FIGURE 13.** **Some examples of forecasts.**

layers, and the dropout probability in the attention layers. Additionally, there are hyperparameters for the maximum number of time steps in positional encoding, as well as the maximum values for the hour, minute, day of week, and total minute counts in the historical data time part. The pseudo-code flow about the model is as follows:

### B. LOSS FUNCTION

Meanwhile, the loss function used in this paper is Mean squared logarithmic error (mean squared logarithmic error), $\hat{y}$ is the predicted value, $y$ is the true value, n is the number of samples, m is the number of cells (Label number), and the goal is to minimize Loss, i.e., to maximize Score. The purpose of designing this indicator as an evaluation function is to penalize the underestimation that exists in water supply forecasting. That is, it is better to overestimate the data than to underestimate them. The expressions are as follows.

$$\begin{aligned} \mathrm{MSLE}(y, \hat{y}) &= \frac{1}{n}\sum_{i=1}^{n}\left(\log_e(1+y_i) - \log_e\left(1+\hat{y}_i\right)\right)^2 \\ \text{Loss } &= \sum_{i=1}^{m}\mathrm{MSLE}\left(y_i, \hat{y}_i\right) \\ \text{Score } &\equiv 1/(1+\text{ Loss }) \end{aligned} \tag{6}$$

#### 1) MODEL PERFORMANCE ANALYSIS

In the original data, there are many obvious problems, such as missing data (large data gaps are missing), obvious data errors (water consumption is negative), obvious abnormal data (serious deviation from the value of the mean variance), in order to analyze the data, it is necessary to carefully clean and organize the data in advance, and some tables do not have data information such as weather, epidemic, etc. The most complete data set is the hourly interval flow data set The most complete data set is the hourly interval flow data set and the daily interval flow data set, so as far as possible, the hourly interval flow data set is used as the basis, and the rest of the data set is used as auxiliary information to build multiple features to assist the prediction and thus improve the prediction effect of the model.

The data collected in this paper has obvious periodicity (such as the first few days of each week, the first few days of each month), and there are obvious fluctuations in the cycle, such as the water consumption of flow_1 in June, which can be clearly seen on weekdays, such as the peak of water consumption in the morning and evening, while the weekend and Sunday will be relatively scattered, taking these into account, some features can be added to assist in prediction, so in the actual prediction process Therefore, in the actual prediction process, we added features such as days to indicate that this is the first day of the month, record the day of the week, and record the hour of the day to integrate more features and make full use of the periodicity of the data. As shown in the figure below, water consumption will have small peaks at e.g. 8:00, 12:00, and 9:00 pm.

In this paper, the MSLE loss function is chosen because it penalizes the underestimation in water supply prediction. In order to ensure a smaller loss as much as possible, the final actual output prediction data can be a little bit larger than the intermediate model output data, so as to ensure that the background of water use will not be under-predicted as much as possible when predicting, which leads to the situation of not enough water.

The model prediction part of this paper takes 4 stages to carry out, where each stage of the loss iterative process and Valid Score score iterative process is shown in the figure 11 and figure 12, respectively, save, need to do test set prediction 4 times. It can be seen that all four modules start to converge sharply when the eoch is 15, and the final loss value is around 1.5 from the original 180, which can be seen that the model as a whole is in a convergence state. Meanwhile, regarding the valid score, it starts from about 0.70 and finally converges to around 0.98, which indicates that the model has a good scientific generalization ability. In conclusion, as the training proceeds, the loss of the model is decreasing, indicating that the model is getting better in the training process. At the beginning of the model training, the loss is large, but as the training proceeds, the loss gradually decreases, indicating that the model is continuously optimized and is getting better and better at fitting the training data. At the later stage of training, the loss de-creases more slowly, indicating that the model has reached a certain stability and is difficult to optimize further. In order to prevent overfitting, dropout layer is added, and the hidden states in each LSTM-Transformer unit are randomly discarded with a certain probability to prevent some neurons in the neural network from fitting a certain part of the training data exclusively, thus preventing overfitting of the model. The training speed of the model reaches up to 1.83step/s, indicating that the model has high efficiency in the training process.

**TABLE 2. Experimental results comparison.**

| Models | Number of test sets | MSE | RMSE | MAE |
|---|---|---|---|---|
| Transformer | 168 | 3.987 | 5.032 | 2.398 |
| | 168 | 3.215 | 5.121 | 2.387 |
| | 168 | 3.341 | 5.112 | 2.312 |
| | 168 | 3.342 | 5.654 | 2.365 |
| LSTM | 168 | 4.217 | 5.032 | 2.398 |
| | 168 | 4.321 | 5.121 | 2.387 |
| | 168 | 3.765 | 5.112 | 2.312 |
| | 168 | 3.957 | 5.654 | 2.365 |
| TCN | 168 | 3.740 | 5.032 | 2.028 |
| | 168 | 3.786 | 5.121 | 2.371 |
| | 168 | 3.817 | 5.112 | 2.209 |
| | 168 | 3.875 | 5.654 | 2.297 |
| BiLSTM | 168 | 3.655 | 5.032 | 2.138 |
| | 168 | 3.519 | 5.121 | 2.110 |
| | 168 | 3.569 | 5.112 | 2.245 |
| | 168 | 3.682 | 5.654 | 2.314 |
| OURS | 168 | 3.337 | 5.032 | 1.932 |
| | 168 | 3.441 | 5.121 | 1.885 |
| | 168 | 3.475 | 5.112 | 1.848 |
| | 168 | 3.421 | 5.654 | 1.990 |

### *C. COMPARATIVE EXPERIMENTS*

According to the results presented in Table 2, it is clear that the LTMFormer model outperforms the other four models when it comes to predicting the demand for secondary water supply in residential communities. Specifically, the LTMFormer model achieved the lowest mean squared error (MSE) of 3.418 among all test sets, which is slightly lower than the Transformer model's MSE value of 3.471 and significantly lower than the other three models. In addition, the LTMFormer model recorded the smallest mean absolute error (MAE) of 1.913 across all test sets, which is also lower than the other four models. Furthermore, the LTMFormer model performed better than the other models in terms of the average root mean square error (RMSE) on all test sets, with a value of 5.480.

These findings suggest that the LTMFormer model offers a practical and effective solution for predicting the demand for secondary water supply in residential communities. The model's superior performance may be attributed to its merging of LSTM and Transformer models with a self-attention structure, which enables the model to capture long-term dependencies and time-series characteristics of the data more accurately.

Overall, this study provides a practical approach for predicting the demand for secondary water supply in residential communities using the LTMFormer model. This model could be a valuable tool for water resource planning and management in various settings, including urban and rural areas. From the above table, we can see that the combined LTMFormer model proposed in this paper has better prediction accuracy and scientific generalization ability compared to similar models. All evaluation indexes are in the best state, which overall conform to the data distribution characteristics. Figures 13a, 13b, 13c, and 13d represent the predicted water supply demand for 20 cells at four different time periods, all of which span over 168 hours.

## V. CONCLUSION

In conclusion, the LTMFormer model proposed in this paper is a novel approach that effectively combines the LSTM and Transformer models to enhance the accuracy and length of the prediction in long-term sequence modeling tasks. The results of our experiments on a metric dataset of 20 cells in Shanghai demonstrate that the LTMFormer outperforms traditional deep learning models such as LSTM, Transformer, TCN, and BiLSTM, achieving the best performance of MSE, RMSE, and MAE of 3.337, 4.536, and 1.848, respectively.

The cyclical changes observed in the prediction examples further highlight the ability of the LTMFormer model to capture complex patterns in the data. Such accurate predictions can help protect the public's water health and meet the growing demand for a better life.

Overall, we believe that the LTMFormer model presents a promising direction for future research in the field of long-term sequence modeling, and its potential applications extend beyond water quality monitoring to other domains such as finance, weather forecasting, and healthcare.

## REFERENCES

[1] H. Vicente, S. Dias, A. Fernandes, A. Abelha, J. Machado, and J. Neves, "Prediction of the quality of public water supply using artificial neural networks," *J. Water Supply, Res. Technol.-Aqua*, vol. 61, no. 7, pp. 446–459, Nov. 2012.

[2] A. Masduqi, "Prediction of rural water supply system sustainability using a mathematical model," *Jurnal Purifikasi*, vol. 10, no. 2, pp. 155–164, 2009.

[3] A. Antunes, A. Andrade-Campos, A. Sardinha-Lourenço, and M. S. Oliveira, "Short-term water demand forecasting using machine learning techniques," *J. Hydroinformatics*, vol. 20, no. 6, pp. 1343–1366, Nov. 2018.

[4] I. S. Msiza, F. V. Nelwamondo, and T. Marwala, "Artificial neural networks and support vector machines for water demand time series forecasting," in *Proc. IEEE Int. Conf. Syst., Man Cybern.*, Oct. 2007, pp. 638–643.

[5] A. Maleki, S. Nasseri, M. S. Aminabad, and M. Hadi, "Comparison of ARIMA and NNAR models for forecasting water treatment plant's influent characteristics," *KSCE J. Civil Eng.*, vol. 22, no. 9, pp. 3233–3245, Sep. 2018.

[6] K. N. Irvine and A. J. Eberhardt, "Multiplicative, seasonal ARIMA models for Lake Erie and Lake Ontario water levels," *J. Amer. Water Resour. Assoc.*, vol. 28, no. 2, pp. 385–396, Apr. 1992.

[7] G. Viccione, C. Guarnaccia, S. Mancini, and J. Quartieri, "On the use of ARIMA models for short-term water tank levels forecasting," *Water Supply*, vol. 20, no. 3, pp. 787–799, May 2020.

[8] P. J. Oliveira, J. L. Steffen, and P. Cheung, "Parameter estimation of seasonal ARIMA models for water demand forecasting using the harmony search algorithm," *Proc. Eng.*, vol. 186, pp. 177–185, Jan. 2017.

[9] M. Kumar and M. Thenmozhi, "Forecasting stock index returns using ARIMA-SVM, ARIMA-ANN, and ARIMA-random forest hybrid models," *Int. J. Banking, Accounting Finance*, vol. 5, no. 3, pp. 284–308, 2014.

[10] Y. Xu, J. Zhang, Z. Long, and Y. Chen, "A new hybrid approach for short-term water demand time series forecasting," in *Proc. 13th World Congr. Intell. Control Autom. (WCICA)*, Jul. 2018, pp. 534–539.

[11] J. Guo, H. Sun, and B. Du, "Multivariable time series forecasting for urban water demand based on temporal convolutional network combining random forest feature selection and discrete wavelet transform," *Water Resour. Manage.*, vol. 36, no. 9, pp. 3385–3400, Jul. 2022.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–11.

[13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.

[14] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16 × 16 words: Transformers for image recognition at scale," 2020, *arXiv:2010.11929*.

[15] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: LSTM cells and network architectures," *Neural Comput.*, vol. 31, no. 7, pp. 1235–1270, Jul. 2019.

[16] W. Bao, J. Yue, and Y. Rao, "A deep learning framework for financial time series using stacked autoencoders and long-short term memory," *PLoS ONE*, vol. 12, no. 7, Jul. 2017, Art. no. e0180944.

[17] P. Delgado-Santos, R. Tolosana, R. Guest, F. Deravi, and R. Vera-Rodriguez, "Exploring transformers for behavioural biometrics: A case study in gait recognition," 2022, *arXiv:2206.01441*.

[18] F. Andayani, L. B. Theng, M. T. Tsun, and C. Chua, "Hybrid LSTM-transformer model for emotion recognition from speech audio files," *IEEE Access*, vol. 10, pp. 36018–36027, 2022, doi: 10.1109/ACCESS.2022.3163856.

[19] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[20] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proc. ICML*, 2013, pp. 1310–1318.

**DALI LI** was born in Tangshan, Hebei, China, in 1980. He is currently with the Division of Asset and Laboratory Management, Shanghai Normal University. His research interests include big date, data mining, and environmental science.

**QINGWEN FU** was born in Shanghai, China, in 1981. He is currently with the College of Information, Mechanical and Electrical Engineering, Shanghai Normal University. His research interests include data mining and communication engineering.