Research on Time Series Forecasting Models Based on Hybrid Attention Mechanism and Graph Neural Networks

Yirui Cheng, Guo Li*, Xu Zhou, Shuhui Ye
School of Artificial Intelligence and Software Engineering, Nanyang Normal University, Nanyang, Henan, 473061, China
E-mail: airforce1980@126.com
*Corresponding author

Keywords: multivariate time series forecasting, transformer, attention mechanism, multi-scale features, convolutional neural networks, graph convolution

Received: November 12, 2024

In wireless data transmission, packet loss and missing data caused by environmental interference and network congestion significantly impact the stability of time series. To address these challenges, this study proposes a time series forecasting model named FGDLNet. FGDLNet is based on the Transformer architecture and integrates Graph Neural Networks (GNN) to enhance the performance of long sequence predictions, particularly in handling complex time series patterns. The model simplifies its structure and reduces computational complexity by removing the Decoder module from the traditional Transformer and replacing it with a linear layer for direct connection and prediction. To enhance the feature extraction capability of time series data, FGDLNet incorporates a multi-scale feature extraction module that extracts features at different temporal scales using multiple convolution kernels in parallel. Specifically, the model employs a single-channel processing approach to reduce interference between features and improve prediction accuracy. The introduced GNN module enables feature propagation and enhancement within the single channel, better capturing short-term fluctuations and long-term trends. In terms of the attention mechanism, this study designs a hybrid attention mechanism that combines global linear attention and local window attention. The global linear attention optimizes the computation to improve the efficiency of capturing global contextual information, while the local window attention strengthens the model's ability to handle short-term dependencies. To evaluate the effectiveness of the model, we selected a dataset recorded during the flight of a specific aircraft, which includes longitude, latitude, and inertial navigation parameters, and conducted long-term trend forecasting. In the experiments, we used MAE (Mean Absolute Error), MSE (Mean Squared Error), and training time to assess the model's performance. The experimental results show that FGDLNet outperforms traditional models such as Autoformer, Transformer, Informer, Reformer, DLinear, and ITransformer in long-term forecasting tasks. Specifically, FGDLNet achieves the following MAE values: 0.1400, 0.0595, 0.0092, 0.0324, 0.0493, and 0.122, which are significantly lower than those of the other models. In terms of MSE, FGDLNet also demonstrates lower errors: 0.0231, 0.0584, 0.0987, 0.0825, 0.0798, and 0.1925. Additionally, FGDLNet's training time per epoch is 1156.47 seconds, which is about 7% faster than the Transformer model (1243.14 seconds).

Povzetek:Predstavljen je FGDLNet model za napovedovanje časovnih vrst, ki združuje transformersko arhitekturo, grafne nevronske mreže in hibridni mehanizem pozornosti za izboljšano napovedovanje dolgih zaporedij.

1 Introduction

In recent years, time series forecasting has gained increasing importance across various fields, particularly in wireless data transmission. Accurate long-term forecasting in time series remains a challenging task due to several inherent difficulties, such as capturing both long-term dependencies and short-term fluctuations, and handling noise and missing data in real-world scenarios. In wireless data transmission, issues like packet loss, network congestion, and signal interference often result in missing data or inaccuracies in measurements, complicating the forecasting process. These challenges are particularly critical in scenarios where accurate, real-time predictions are needed to ensure reliable system performance. Furthermore, traditional time series models often struggle with efficiently capturing complex, high-dimensional data patterns, especially when multiple dependencies across time steps need to be modeled.

To address these issues, this paper introduces FGDLNet, a novel long-sequence time series forecasting model. In this paper, we introduce FGDLNet, a novel time series forecasting model designed to handle long sequences while addressing the challenges of capturing both global and local dependencies. The name FGDLNet reflects the core design principles of the model:

F stands for Feature Extraction, emphasizing the model multi-scale feature extraction module that captures key features at different temporal scales.

G represents Graph Neural Networks, which enhance the model ability to capture complex dependencies between time steps and improve performance, particularly in noisy or high-dimensional settings.

D refers to Decoder Removal, signifying the simplification of the Transformer architecture by removing the decoder, reducing computational complexity.

L stands for Local and Global Attention, highlighting the hybrid attention mechanism that combines local window attention and global linear attention to effectively model both short-term fluctuations and long-term trends.

Net indicates that it is a Network, emphasizing the use of a deep learning framework based on Transformer architecture.

2 Related works

Time series forecasting has long been a critical research area, with numerous methods proposed to model temporal dependencies in data. The Transformer model, due to its ability to effectively capture long-range dependencies through attention mechanisms, has become a key advancement in sequence modeling tasks [1]. However, its computational complexity grows quadratically with sequence length, limiting its scalability and efficiency in handling long sequences. Several modified Transformer-based models have been proposed in recent years to address this issue, aiming to reduce computational costs while improving the modeling capability of time series data.

For example, Informer introduces the ProbSparse attention mechanism to improve computational efficiency in long-sequence prediction tasks [2]. While it enhances computational efficiency, it struggles with capturing fine-grained local features, especially when short-term dependencies are critical. Reformer further reduces computational complexity by using locality-sensitive hashing and reversible residual networks [3]. However, it still struggles with handling highly non-linear patterns, common in real-world time series data.

Autoformer and DLinear have also contributed to trend prediction. Autoformer uses a recursive mechanism to enhance its ability to capture long-term trends but faces limitations in modeling short-term dependencies, especially in environments with high volatility and noise [4]. DLinear, using linear decomposition, handles time series data but lacks flexibility in dealing with complex non-linear trends or abrupt changes [5].

Despite these advancements, existing models still face challenges in effectively capturing both long-term global dependencies and short-term local patterns in time series data. Additionally, they often struggle with computational efficiency, particularly when processing high-dimensional, noisy, and large-scale data. To address these limitations, this paper proposes the FGDLNet model, integrating global linear attention for capturing long-term trends, local window attention for short-term fluctuations, a multi-scale feature extraction module for richer feature representation, and a Graph Neural Network module to enhance the model's ability to learn complex dependencies. Through these innovations, FGDLNet not only addresses the limitations of long-sequence forecasting but also demonstrates significant advantages in capturing complex dependencies within time series data. The comparison between algorithms is shown in Table 1.

Table 1: Comparison of the different types of protocols involved.

Algorithm	Vantage	Drawbacks
Autoformer	Automatically captures periodic changes in time series. Improves prediction accuracy.	May require a large amount of data for training. May not perform well with non-periodic data.
Transformer	Efficient parallel computing capability. Strong ability to handle long sequences. Self-attention mechanism captures complex dependencies.	High computational resource consumption. May encounter efficiency issues with very long sequences.
Informer	Designed for long sequence time series forecasting. ProbSparse self-attention mechanism reduces computational complexity.	May not be as effective for short sequences compared to traditional Transformer. Higher implementation complexity.
Reformer	Reduces computational complexity with LSH attention mechanism. Reversible network reduces memory consumption.	LSH may introduce approximation errors. May not be flexible enough for some tasks.
DLinear	Simple and efficient time series forecasting model. O(1) maximum signal traversal path length. Consumes less memory and parameters.	May not be suitable for all types of time series data. Limited ability to capture complex patterns.

3 Methodology

In this paper, we introduce FGDLNet, a time series prediction model based on the Transformer architecture. The input time series data is first processed through an embedding layer, which maps the data to a high-dimensional feature space to capture temporal features more effectively. This embedding process is similar to the embedding operation in a standard Transformer.

To handle time series data with multi-scale features, we designed a multi-scale feature extraction module. This module applies multiple sets of one-dimensional convolution operations to extract features at different temporal scales. Each convolutional kernel captures dependencies within a specific temporal range, resulting in multi-scale feature representations. These features are then combined and fed into the attention mechanism module.

The attention mechanism includes a global linear attention mechanism to optimize computation, significantly reducing complexity and enabling efficient handling of long-sequence data while retaining global contextual information. This helps the model understand overall trends in the data. To capture local patterns more effectively, we introduced a local window attention mechanism, which focuses on sequence segments within a fixed-size window. This approach reduces computational complexity and enhances the model's ability to capture local features.

We also integrated a Graph Neural Network module to enhance FGDLNet's ability to capture non-linear dependencies in time series data, improving accuracy, especially in noisy scenarios. Additionally, by using a single-channel processing approach, we reduce feature interference and avoid the complexity of multi-channel processing, ensuring better feature representation and improving the model's stability and accuracy in time series forecasting.

At the core of FGDLNet are multiple stacked encoder layers, each comprising a global linear attention module, a local window attention module, and a GNN module. By stacking these layers, the encoder captures both long- and short-term dependencies in the time series data. The GNN module further enhances the handling of complex dependencies within the data, improving predictive capability. The feature representations output by the encoder layers are transformed through a linear transformation layer to produce the final prediction result. The output layer design ensures that the model can generate accurate time series predictions based on the captured global and local features and node dependencies. The FGDLNet framework is illustrated in Figure 1.

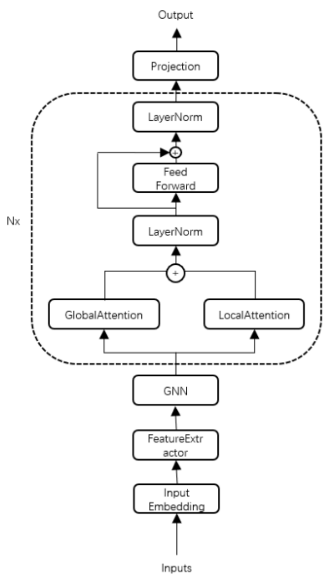

Figure 1: Network structure model.

3.1 Feature engineering

In this study, we propose a time-stamp-based feature extraction method to enhance the model's ability to capture periodic and seasonal patterns in time series data. Compared to traditional methods, we further enrich the diversity and granularity of time features, particularly when handling high-frequency time series data with second-level resolution. Specifically, we dynamically generate corresponding time features based on the time frequency of the input data (such as seconds, minutes, hours, etc.), and encode them by combining multiple time units [6].

Unlike traditional time features (such as seconds, minutes, hours, and dates), our method extends to more refined features, such as decomposing minutes into different time intervals (e.g., 2 minutes, 10 minutes, and 20 minutes) for feature encoding. These interval-based features capture micro-level time variations within the time series, which are particularly useful for high-frequency data (e.g., per-second time series) as they help identify finer-grained periodic patterns. To ensure that all time features are within a unified scale, we normalize their values to the range of $[-0.5, 0.5]$, avoiding training instability caused by scale differences. Ultimately, all of these time features are stacked into a feature matrix, serving as the model's input. This multi-level, multi-granular approach to time features helps the model better capture the periodicity and regularity in time series data, thus improving prediction performance [7-9].

In summary, compared to traditional time feature extraction methods, our extended approach refines time units and adds multiple time interval features, enabling the model to comprehensively capture various time scales and periodic patterns in the data. This significantly enhances the model's performance, particularly when handling high-frequency time series data.

3.2 Multiscale feature extraction

In the time series forecasting model, the multi-scale feature extraction module is designed to capture information across various temporal scales, aiming to provide enriched feature representations for more accurate predictions. The data is recorded every second, with four feature values that capture periodic behavior and patterns in the time series. By encoding the minute values in various ways, the model can recognize and leverage cyclical patterns effectively. Time features are extracted at scales of every hour, every minute, every second, two-minute intervals, ten-minute intervals, and twenty-minute intervals.

Before feature extraction, the time series data's distinct features are separated into individual channels. This feature separation approach ensures that convolution operations within each channel focus on a specific feature, preventing interference among features. As a result, the model can more accurately capture the independent patterns of each feature [10-12].

To capture features across multiple temporal scales, the multi-scale feature extraction module applies various convolutional kernels in parallel. Specifically, different kernel sizes (such as 3, 5, and 7) are used to convolve each channel's data. Smaller kernels (e.g., 3x3) are adept at capturing short-term dependencies, while larger kernels (e.g., 7x7) cover broader context, capturing long-term dependencies. This multi-scale convolution operation enables the model to extract layered feature representations from the time series, enriching the information available for subsequent processing. These time features are then processed and fed into the model, as shown in the data processing flow in Figure 2.

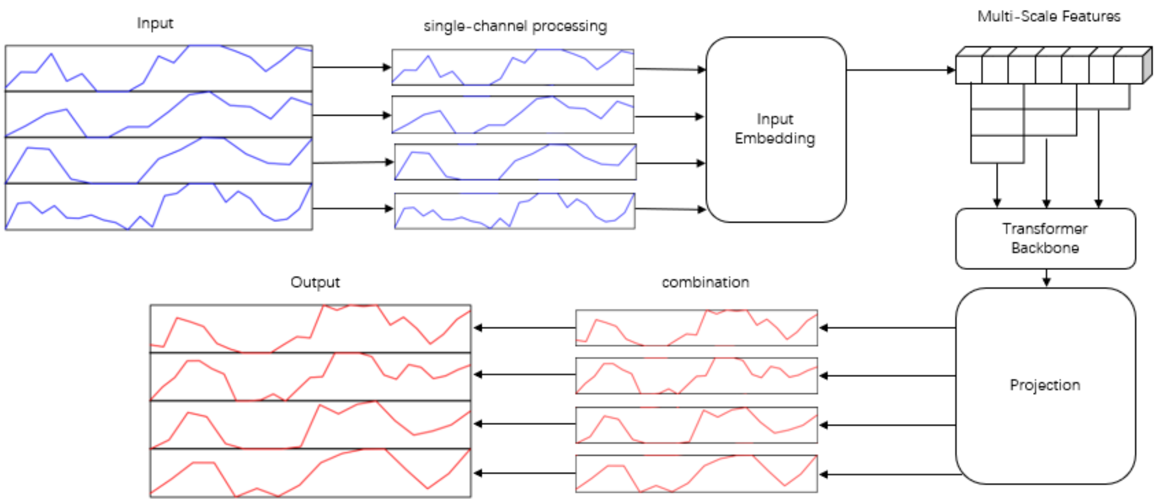

Figure 2: Data processing procedure

Because different kernel sizes generate feature maps of varying lengths, it's necessary to align these feature maps before integration. To achieve this, the module pads each feature map to a uniform length, ensuring consistent shapes for concatenation. After alignment, feature maps from all channels are concatenated along the feature dimension, forming a composite feature representation.

An embedding layer then transforms the concatenated features, enhancing the representational capacity. This embedding layer acts as a linear transformation, mapping the multi-scale features into a unified feature space for processing by subsequent encoder layers.

Finally, the integrated multi-scale features are processed through a fully connected layer to form the model's final input. Although each channel processes different features independently, the fusion process allows the model to capture interrelationships among these features.

In summary, the multi-scale feature extraction module leverages feature separation, multi-scale convolution, feature alignment and concatenation, and embedding transformation to extract and integrate rich information across different temporal scales in the time series. This design improves the model's capacity to capture both short-term and long-term dependencies, enhancing FGDLNet's predictive performance on complex time series data.

Given an input time series $X \in \mathbb{R}^{B \times L \times D}$, where B is the batch size, L is the sequence length, and D is the feature dimension, each feature $x_d \in \mathbb{R}^{B \times L}$is separated into individual channels and then input into multi-scale convolutional layers:

$$X_1^{(d)} = \mathrm{Conv1d}(x_d, k_1) + \mathrm{ReLU} + \mathrm{BN} \quad (1)$$

$$X_2^{(d)} = \mathrm{Conv1d}(x_d, k_2) + \mathrm{ReLU} + \mathrm{BN} \quad (2)$$

$$X_3^{(d)} = \mathrm{Conv1d}(x_d, k_3) + \mathrm{ReLU} + \mathrm{BN} \quad (3)$$

where k_1,k_2,k_3 represent different kernel sizes used for capturing features across various temporal scales.

The output of these convolutional layers can be denoted as:

$$X^{(d)} = \mathrm{Concat}([X_1^{(d)}, X_2^{(d)}, X_3^{(d)}]) \quad (4)$$

The final integrated features are further processed through a linear layer:

$$X'^{(d)} = \mathrm{Linear}(X^{(d)}) \quad (5)$$

where $X'^{(d)}$is the final integrated output, which serves as the input to the subsequent attention mechanism. This enables FGDLNet to capture multi-scale feature information, enhancing both the accuracy and stability of its predictions.

3.3 Global linear attention

In the time series prediction model proposed in this paper, the global linear attention mechanism is one of the key components, aimed at capturing global contextual information in time series data while maintaining efficient computation. Traditional self-attention mechanisms perform well in handling long sequence data, but their computational complexity is $O(N^2)$, where N represents the sequence length. When processing large-scale long sequence data, this

high computational complexity leads to a sharp increase in computational and memory overhead, which impacts the efficiency and scalability of the model. To address this issue, this paper introduces the global linear attention mechanism.

The core idea of global linear attention is to capture global contextual information while using a more computationally efficient attention calculation method.

Specifically, global linear attention replaces the dot-product operation in the traditional self-attention mechanism with an operation that has linear time complexity [13]. This allows the global linear attention mechanism to maintain high computational efficiency when handling long sequences.

In the global linear attention mechanism, the attention calculation at each time step no longer depends on all other time steps. Instead, it aggregates the information from all-time steps in the sequence through a linear combination. This approach enables FGDLNet to capture the overall patterns of the sequence during the attention calculation and avoids the high complexity of calculating attention scores for each time step, as in traditional self-attention mechanisms [14].

After the input sequence passes through the embedding layer, it is mapped to a high-dimensional feature space. Each time step in the time series is represented as a feature vector, Q_i , K_i , V_i with the query (Query), key (Key), and value (Value) corresponding to the respective components.

In the global linear attention mechanism, the calculation of attention weights no longer depends on all time steps. Instead, it aggregates the information from the entire sequence through a linear computation. Specifically, this can be expressed by the following formula:

$$QK = \frac{Q \cdot K^T}{\sqrt{d_k}} \tag{6}$$

In this context, $Q \in \mathbb{R}^{B \times L \times H \times E}$, and $K \in \mathbb{R}^{B \times S \times H \times E}$ represent feature dimensions, and d_k is the scaling factor. The computed QK is then passed to the SoftMax layer to generate the attention weights A:

$$A = \text{Softmax}\left(\frac{QK}{\sqrt{d_k}}\right) \tag{7}$$

The global linear attention mechanism captures global information efficiently through the formula above. The attention weights A reflect the similarity between the query and the key, thereby capturing global patterns in the sequence. Next, by multiplying these weights with the values V, new feature representations are generated:

$$Output = A \cdot V \tag{8}$$

By using a linear time complexity calculation method, the global linear attention mechanism avoids computational and memory bottlenecks when processing long sequences, allowing the model to handle longer sequences and significantly improving computational efficiency.

3.4 Local window attention

The core idea of local window attention is to divide the time series into fixed-size windows and calculate attention weights within each window. This approach avoids the pairwise computation of attention scores for all time steps in the sequence, as in global attention mechanisms, significantly reducing the computational complexity. It is particularly well-suited for handling long sequence data [15-16].

The length of the input sequence is L and the feature dimension is d with a window size of W. In each window, the specific calculation steps for local window attention are as follows:

First, the input sequence is divided into windows of size W Each window contains W time steps, and the feature at each time step is represented as a vector.

Attention Weight Calculation, for each time step i and j within the window (where $i,j \in [1, W]$ represents the range of time steps in the window), the dot product between the query Q and the key K is computed to obtain the attention score matrix:

$$QK_{i,j} = Q_i \times K_j^T \tag{9}$$

Where Q_i and K_j represent the query and key vectors of the i and j time steps within the window, respectively.

Scaling and Normalization, the attention score matrix is scaled by the feature dimension d and then the SoftMax operation is applied to the results:

$$A_{i,j} = \text{softmax}\left(\frac{\mathrm{QK}_{i,j}}{\sqrt{d}}\right) \tag{10}$$

Where $A_{i,j}$ represents the attention weight between the i and j time steps within the window.

The attention weight matrix A is then used to perform a weighted sum on the value matrix V to generate the output features within the window:

$$Output_i = \sum_{j=1}^{W} A_{i,j} \times V_j \tag{11}$$

Where V_j is the value vector of the j time step within the window.

The output features of all windows are concatenated sequentially to form the final output sequence. For the entire sequence, the output of the local window attention mechanism O can be represented as:

$$O = \text{Concat}\left(\left[\sum_{j=1}^{W} \text{softmax}\left(\frac{Q_i \times K_j^T}{\sqrt{d}}\right) \times V_j\right]_{i=1}^{W}, \dots, \left[\sum_{j=1}^{W} \text{softmax}\left(\frac{Q_i \times K_j^T}{\sqrt{d}}\right) \times V_j\right]_{i=L-W+1}^{L}\right) \tag{12}$$

Where concatenate indicates the concatenation of the output features of all windows in sequence order.

The window size W plays a decisive role in the local window attention mechanism as it limits the range of time steps considered during each attention weight calculation. By selecting W appropriately, local patterns in the time series can be captured while ensuring computational efficiency. This mechanism is particularly well-suited for processing long sequence data, effectively balancing the capture of both global and local information.

3.5 Mixed attention mechanism

In this paper, we propose a hybrid attention mechanism designed to effectively combine global and local attention to capture both long-range and short-range dependencies in time series data. The aim of this mechanism is to balance the model's ability to understand global temporal patterns while maintaining computational efficiency. In traditional attention mechanisms, global dependencies are captured by attending to all time steps across the entire sequence, while local dependencies are often overlooked or poorly represented [17]. To overcome this challenge, we introduce a mechanism that integrates both types of attention, ensuring that the model can simultaneously focus on local interactions and global temporal relationships.

The key idea behind this hybrid approach is to decouple the computations of global and local attention, which are then performed in parallel and combined afterward. To integrate these two types of attention, we apply them to different layers or parallel branches of the model. The global attention mechanism operates over the entire sequence, aggregating information from all-time steps to capture long-term trends, while the local attention mechanism focuses on a sliding window of time steps to capture local patterns.

The outputs of these two attention mechanisms are then integrated through a weighted summation. This process can be formulated as follows:

The global linear attention mechanism generates feature representations that capture sequence-level temporal dependencies. This is represented as:

$$\text{Global Output} = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right) \cdot V \qquad (13)$$

where Q represents the queries, K the keys, and V the values, with d_k being the dimensionality of the keys (and queries).

The local window attention mechanism generates feature representations focused on specific time windows. This can be expressed as:

$$\text{Local Output} = \sum_{i=1}^{M} softmax\left(\frac{Q_i K_i^T}{\sqrt{d_k}}\right) \cdot V_i \qquad (14)$$

where Q_i, k_i, and V_i refer to the queries, keys, and values within the i window of the sliding window, and M is the total number of windows.

Finally, the outputs of the global and local attention mechanisms are combined. This integration is performed through weighted summation:

$$\text{Combined Output} = \text{Global Output} + \text{Local Output} \qquad (15)$$

This approach enables the model to balance the capture of global patterns and local details, ensuring a more comprehensive understanding of the time series data while maintaining computational efficiency.

3.6 Graph convolution module

This paper proposes the introduction of a Graph Neural Network module in time series forecasting models. GNN leverages graph structures to represent the relationships between nodes in time series data and learns complex dependencies in the sequence through information propagation between nodes [17]. Specifically, we treat each time step or feature in the time series data as a node in the graph, where the connections between the nodes represent the dependencies between these time steps or features. The GNN module uses multi-layer graph convolution operations to map these node features into a new feature space, capturing both global and local dependencies.

The input time series data is represented as a three-dimensional tensor $X \in \mathbb{R}^{B\times L\times F}$, where B represents the batch size, L represents the sequence length, and F represents the feature dimension.

Another input is the adjacency matrix $A \in \mathbb{R}^{L\times L}$ or edge list E, which represents the connections between nodes. For each graph convolution layer, the feature update of node l in the i layer is expressed as:

$$h_i^{(l)} = \sigma\left(\sum_{j\in\mathcal{N}(i)} \frac{1}{\sqrt{|\mathcal{N}(i)|\cdot|\mathcal{N}(j)|}} W^{(l)} h_j^{(l-1)} + b^{(l)}\right) \quad (16)$$

Here, $h_i^{(l)}$ is the feature representation of node l in the i layer, $\mathcal{N}(i)$ is the set of neighboring nodes of node i, c_{ij} is the normalization coefficient used for scaling, $w^{(l)}$ and $b^{(l)}$ are the weight matrix and bias term, respectively, and $\sigma(\cdot)$ is the activation function. $\frac{1}{\sqrt{|\mathcal{N}(i)|\cdot|\mathcal{N}(j)|}}$ is the normalization coefficient to ensure the numerical stability of the features.

In the feature extraction module, local features of the time series are first extracted through convolutional layers:

$$X^{\text{conv}} = \text{Conv1d}(X) \qquad (17)$$

Then, the convolved features are combined with the adjacency matrix and input into the GNN layer to perform graph convolution operations, generating new feature representations:

$$H^{gnn} = \text{GNN}(X^{conv}, A) \qquad (18)$$

Finally, the features are further processed through a fully connected layer to obtain the final feature representation:

$$X^{out} = \text{Linear}(H^{gnn}) \qquad (19)$$

Through these formulas, the GNN module can effectively model the complex dependencies in time series data, especially when the data exhibits network structure characteristics. This design enables the model to capture both global and local temporal dependencies, thereby improving prediction accuracy.

In the FGDLNet model presented in this paper, the GNN module processes the graph-structured representation of the input data through graph convolution layers. Specifically, the time series data is treated as a graph, where each time step or feature is a node, and the connections between nodes represent their dependencies.

3.7 Loss function

This paper proposes an improved version of the Mean Squared Error (MSE) loss function, which aims to enhance local dependencies by introducing a penalty term, thereby improving the robustness and accuracy of time series forecasting [18-20]. The loss function consists of two components: the standard MSE loss and a penalty term. The penalty term is used to penalize the variation in prediction errors between adjacent time steps, ensuring the stability of the predictions.

$$\mathcal{L}_1 = \frac{1}{N}\sum_{i=1}^{N}(pred_i - true_i)^2 \tag{20}$$

The penalty term measures the difference in prediction errors between adjacent time steps, penalizing large changes in the prediction errors to ensure stability. The formula is as follows:

$$\mathcal{L}_2 = \frac{1}{N}\sum_{i=1}^{N}(|pred_{i+1} - pred_i| - |true_{i+1} - true_i|)^2 \tag{21}$$

The total loss is the sum of the MSE loss and the penalty term, with the hyperparameter λ controlling the balance between the two:

$$\mathcal{L} = \mathcal{L}_1 + \lambda \cdot \mathcal{L}_2 \tag{22}$$

Compared to the traditional MSE loss, the penalty term introduced in this paper helps reduce the fluctuations in prediction errors between adjacent time steps, thereby improving prediction stability.

4 Experimental and data analysis

4.1 Dataset selection

The dataset contains recorded information of a specific aircraft model during flight, with data recorded once per second. The fields in the dataset include time, longitude, latitude, true heading (from inertial navigation), and groundspeed. The dataset includes flight records from four different flights for training and prediction. 80% of the dataset is used as the training set, and 20% is used as the test set.

4.2 Experimental setup

Since the performance of deep learning models is highly dependent on the choice of hyperparameters, we refer to the parameter settings of similar models in the literature to define the ranges of the parameters for tuning. The goal is to identify the optimal combination of hyperparameters that leads to the best model performance during training.

The choice of kernel sizes (3, 5, 7) for the convolutional layers is made to capture multi-scale features of the time series data. Smaller kernel sizes, like 3, are designed to capture fine-grained, local patterns, which are crucial for detecting short-term fluctuations in the data. Larger kernel sizes, like 5 and 7, allow the model to capture more global or longer-term dependencies, which are especially important for understanding broader trends and periodic behaviors in time series data. By using a combination of kernel sizes, the model can simultaneously capture both local and global features, improving its robustness and prediction accuracy.

The lookback window size of 384 is chosen to balance capturing long-term dependencies and ensuring computational efficiency. A window of this size allows the model to leverage sufficient historical data (384 seconds) to recognize important trends and fluctuations in the wireless data transmission, without introducing excessive noise or increasing computational load.

The model parameters and their corresponding values used in our experiments are shown in Table 2.

Table 2: Model parameters

parameter name	parameter value
activation function	GELU
optimizer	Adam
learning rate	0.0001
loss function	MSE
Dropout	0.05
sliding window	5
lookback window size	384
prediction length	96
convolutional multi-scale	{3,5,7}
batch size	32

4.3 Comparative analysis of methods

A comparison is made between FGDLNet and the time series forecasting models Autoformer, Transformer, Informer, Reformer, DLinear, and ITransformer. All models are tested using the same dataset and evaluation criteria. The original dataset undergoes cleaning and normalization to ensure data quality and consistency of inputs across models. The predictive performance of each model is evaluated on the test set, and a unified evaluation standard is used for comparison.

4.4 Evaluation metrics

This study utilizes the following evaluation metrics to comprehensively assess the predictive performance of the models: Mean Absolute Error (MAE), Mean Squared Error (MSE), Coefficient of Determination (R^2), Mean Absolute Percentage Error (MAPE), p-value, 95%

Confidence Interval (CI), and Dynamic Time Warping (DTW).

MAE and MSE measure prediction errors, with MSE being more sensitive to outliers. R^2 evaluates the proportion of variance explained by the model, while MAPE expresses errors as a percentage, ensuring interpretability across datasets. Statistical significance is assessed using the p-value and 95% CI, which provide insights into reliability and uncertainty. DTW evaluates the alignment between predicted and actual time series, capturing temporal shifts and dynamic patterns. Together, these metrics provide a comprehensive framework to evaluate both the accuracy and reliability of the models for time series forecasting.

4.5 Experimental results and analysis

The true heading (from inertial navigation) is selected as the feature for long-term forecasting. A comparison of the predicted values and the true values is shown in Figure 3.

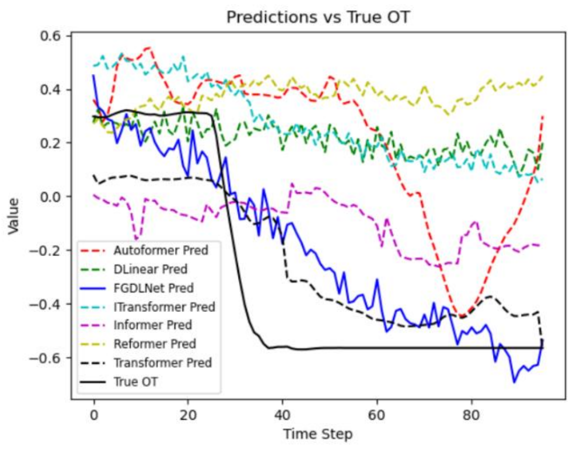

Figure 3: Experimental results

From the fitting plot, it can be observed that the Autoformer model exhibits significant fluctuations throughout the time series. Although it captures some early trends, it deviates considerably from the true values in the later stages, especially with a noticeable overestimation in its predictions. The DLinear model shows relatively stable performance in the early stages, with some consistency with the true values. However, it exhibits a clear downward trend in the middle phase and fails to capture the rebound in the true values. The Transformer model's predictions remain relatively stable but fall within the range of 0.2 to 0.4, which is largely unrelated to the actual downward trend. Similar to Transformer, the Informer model shows stable predictions in the early phase, but then sharply declines, missing the later recovery in the true values. Its overall trend is significantly different from the actual data. The Reformer model's predictions remain high with slight fluctuations throughout the time period, failing to capture the clear downward trend in the true data. FGDLNet successfully identifies the downward trend and fits the predicted data accordingly, achieving lower error for long-term forecasts. This demonstrates the effectiveness of the proposed method, though further model adjustments are needed to balance trend recognition with accuracy in trend amplitude.

The analysis of the MAE, MSE, R^2, and MAPE metrics for the Autoformer, Transformer, Informer, Reformer, DLinear, ITransformer, and FGDLNet models is shown in Table 3.

Table 3: Model evaluation

Model	MAE	MSE	R^2	MAPE
Autoformer	0.2655	0.2723	**0.5219**	1.7205
Transformer	0.1255	0.1029	0.3208	0.7599
Informer	0.1850	0.1382	0.3717	0.8061
Reformer	0.1942	0.1785	0.4225	1.3307
DLinear	0.1618	0.1623	0.4028	**0.3988**
ITransformer	0.2111	0.1596	0.3995	1.3346
FGDLNet	**0.0885**	**0.0798**	0.2825	0.5396

Based on the results in Table 3, this paper analyzes the performance of seven different models in signal prediction, comparing their performance across various metrics including Mean Absolute Error, Mean Squared Error, Coefficient of Determination, and Mean Absolute Percentage Error.

The Transformer model performs well in terms of MAE, with a value of 0.1255, and also shows reasonable performance in MSE, with a value of 0.1029. Its R^2 value is 0.3208, indicating a certain degree of fitting capability. In terms of MAPE, the Transformer achieves an error of 0.7599, which is better than some traditional models but is still outperformed by the FGDLNet model.

In comparison, the Autoformer model shows relatively average performance, with an MAE of 0.2655, an MSE of 0.2723, and an R^2 value of 0.5219, indicating weaker fitting ability. Additionally, its MAPE of 1.7205 highlights its poorer error control performance. The Informer model outperforms Autoformer in terms of MAE and MSE, with values of 0.1850 and 0.1382, respectively, but falls short compared to Transformer. Its R^2 is 0.3717, and its MAPE is 0.8061. The performance of the Reformer model is slightly better than Informer but still does not match Transformer or FGDLNet, with an MAE of 0.1942, an MSE of 0.1785, an R^2 of 0.4225, and a MAPE of 1.3307.

The DLinear model achieves an MAE of 0.1618, an MSE of 0.1623, an R^2 of 0.4028, and a MAPE of 0.3988, demonstrating better performance compared to the traditional models mentioned above. However, it still lags behind FGDLNet. Additionally, the ITransformer model has an MAE of 0.2111, an MSE of 0.1596, an R^2 of 0.3995, and a MAPE of 1.3346. While it performs well in terms of the MSE metric, its overall performance is still inferior to DLinear and Transformer.

The proposed FGDLNet prediction model exhibits significant advantages across all metrics, with an MAE of 0.0885, an MSE of 0.0798, an R² of 0.2825, and a MAPE of only 0.5396. Compared to other models, FGDLNet demonstrates outstanding performance in terms of prediction accuracy and error control. Its exceptional performance fully demonstrates its practical value in addressing issues related to data loss and prediction filling in wireless transmission scenarios.

Based on the results from Figure 4, we analyzed the performance of different models under various prediction horizons and identified significant differences.

To evaluate the impact of different prediction horizons on the performance of the benchmark and prediction models, we choose 384 as the look-back window size L, with prediction lengths $\tau \in \{16,32,48,64,80,96\}$. The results indicate that the proposed model outperforms all baseline models across all datasets. The MAE and MSE prediction results for different prediction horizons are shown in Figure 4.

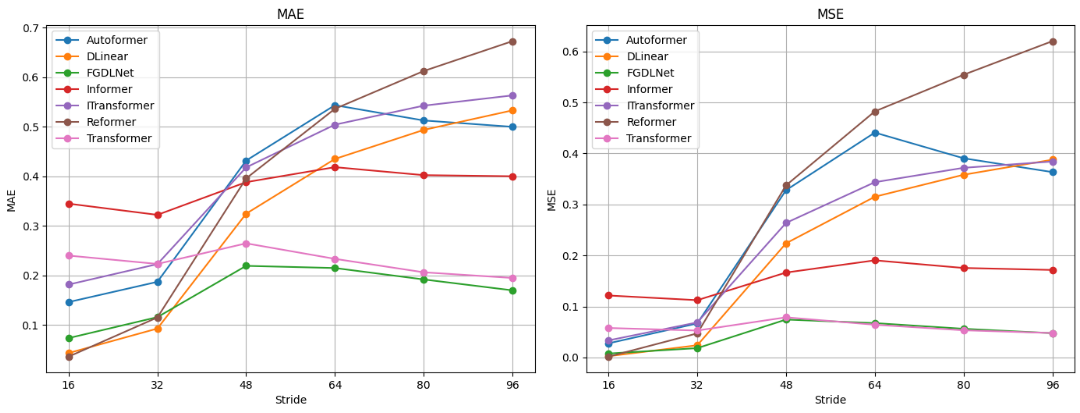

Figure 4: Prediction results

In terms of Mean Absolute Error (MAE), the Autoformer model performs well at shorter prediction horizons (16 and 32). However, as the prediction length increases, the MAE rises significantly, indicating instability in long-term predictions. A similar trend is observed with Mean Squared Error (MSE), where Autoformer shows a substantial increase in error at longer prediction horizons (48 and beyond), highlighting its limitations for handling long-term predictions.

The DLinear model shows excellent performance with shorter horizons (16 and 32), maintaining low MAE and MSE values. Even when the prediction horizon increases to 48 and 64, the error remains relatively stable. For larger prediction horizons (80 and 96), there is a slight increase in error, but overall performance remains superior to other traditional models, indicating DLinear’s good adaptability for long-term predictions.

The Informer model exhibits relatively consistent performance across all prediction horizons, but with higher overall error. At horizons of 48, 64, and beyond, the prediction error increases significantly, suggesting its struggle with long-term forecasting. The MSE follows a similar trend, showing higher error at larger horizons, indicating potential limitations for long-term predictions.

The ITransformer model performs well for shorter horizons (16 and 32), with good MAE and MSE results. However, as the prediction horizon increases, particularly at 48 and beyond, the error grows significantly, revealing poor adaptability to long-term predictions. At the 96-step horizon, the error is notably higher, indicating significant shortcomings when handling longer time spans.

In contrast, FGDLNet consistently performs excellently across all prediction horizons, particularly at longer horizons (80 and 96), where its MAE is significantly lower than that of other models. FGDLNet also excels in MSE, significantly surpassing other models at all prediction horizons, especially at 64, 80, and 96 steps. These results highlight FGDLNet’s advantage in handling long-term prediction tasks.

The Reformer model performs well at shorter horizons (16), with very low MAE. However, as the horizon increases, its error grows substantially, particularly at 64 steps and beyond, showing a clear decline in performance. The MSE for Reformer follows a similar trend, with significant increases in error at longer horizons, indicating poor adaptability for long-term predictions.

The Transformer model exhibits relatively stable performance across all horizons. While MAE and MSE increase slightly at certain prediction horizons (e.g., 48), the overall error remains moderate. Even at larger horizons (80 and 96), Transformer maintains relatively good performance, showcasing its stability when handling predictions at different time steps, but its predictive accuracy lags behind FGDLNet.

In summary, the proposed FGDLNet model demonstrates competitive performance in both short-term and long-term prediction tasks. Particularly in long-term forecasting, its MAE and MSE are significantly lower than those of other models. In comparison, traditional models generally perform poorly with long-term predictions, with errors increasing significantly as the prediction horizon expands. These results highlight the effectiveness and advantages of FGDLNet in wireless data transmission prediction.

4.6 Advanced metrics

In addition to the commonly used MSE, MAE, and R², this study introduces several additional evaluation metrics to comprehensively analyze the performance of different models. These additional metrics include p-value, 95% Confidence Interval, and Dynamic Time Warping. These metrics help evaluate the model's prediction ability, stability, and adaptability to complex time series data, especially when faced with noise, interference, and challenges across different time scales. Table 4 shows the performance of each model in terms of p-value, 95% confidence interval, and dynamic time warping:

Table 4: Model evaluation under high-dimensional metrics

Model	**p-value**	**95% CI**	**DTW**
Autoformer	0.0000	[0.4277, 0.5682]	2.2186
DLinear	0.0000	[0.4336, 0.5799]	5.1743
FGDLNet	0.0000	[0.0595, 0.1391]	0.4831
Informer	0.0000	[0.1218, 0.2707]	3.0734
ITransformer	0.0000	[0.5130, 0.6146]	5.0021
Reformer	0.0000	[0.5817, 0.7459]	7.2481
Transformer	0.0077	[0.0121, 0.0989]	1.2221

4.6.1 p-value and 95% confidence interval analysis

p-value and 95% Confidence Interval are two key metrics for evaluating the significance and stability of model predictions. Specifically, p-value is used to test whether there is a significant difference between the model’s predictions and the actual values. A lower p-value indicates stronger predictive power. The 95% Confidence Interval reflects the consistency and stability of the model's predictions; a narrower interval suggests more consistent and reliable predictions.

In this study, we calculated the p-value and 95% Confidence Interval for each model. The results show that all models have a p-value close to 0.0000, indicating strong predictive power in most cases. However, it is noteworthy that the 95% Confidence Interval of the FGDLNet model is the narrowest ([0.0595, 0.1391]), suggesting that its predictions are more stable and consistent compared to other models.

4.6.2 Dynamic time warping metric analysis

Dynamic Time Warping is a metric used to measure the alignment of two time series along the time axis. It is commonly used to assess the performance of models in time series forecasting tasks. The smaller the DTW value, the higher the similarity between the predicted and actual time series, indicating better model performance.

From the experimental results, FGDLNet performs the best, achieving the lowest DTW value (0.4831) among all models. This low value indicates that FGDLNet aligns most closely with the actual data’s time series, meaning it captures the dynamic patterns of the real data with high accuracy.

4.6.3 Residual plot analysis

In addition to the p-value, 95% confidence interval, and DTW, the residual plot also provides valuable insights into the model's prediction accuracy. The residual plot visualizes the difference (residuals) between the predicted values and the actual observations at different time steps. This is crucial for detecting error patterns that the model may have failed to capture.

A model’s residual plot indicates its ability to capture the underlying dynamics of the data and whether there are any systematic errors. Ideally, the residuals should be randomly distributed around zero, suggesting that the model has captured all the predictable patterns in the data. Figure 5 shows a comparison of the residual plots for the FGDLNet model and the Transformer model. By comparing the residual plots of the two models, it is clear that FGDLNet outperforms the Transformer model in terms of prediction accuracy and stability. The residual points of FGDLNet are closer to the zero-residual line and are more evenly distributed, indicating that it better captures the dynamic variations in the time series data. While the Transformer model

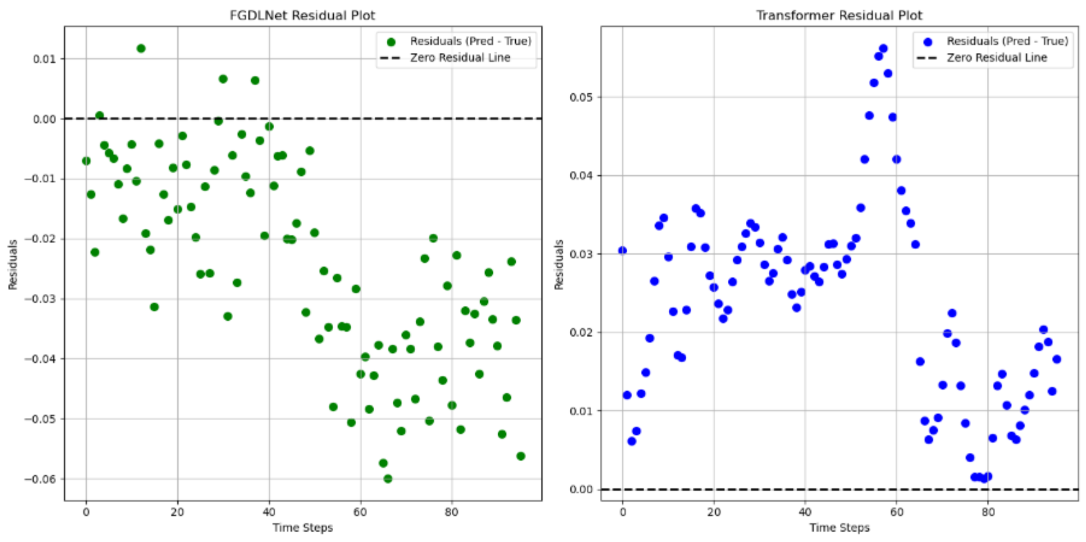

Figure 5: Residual plot

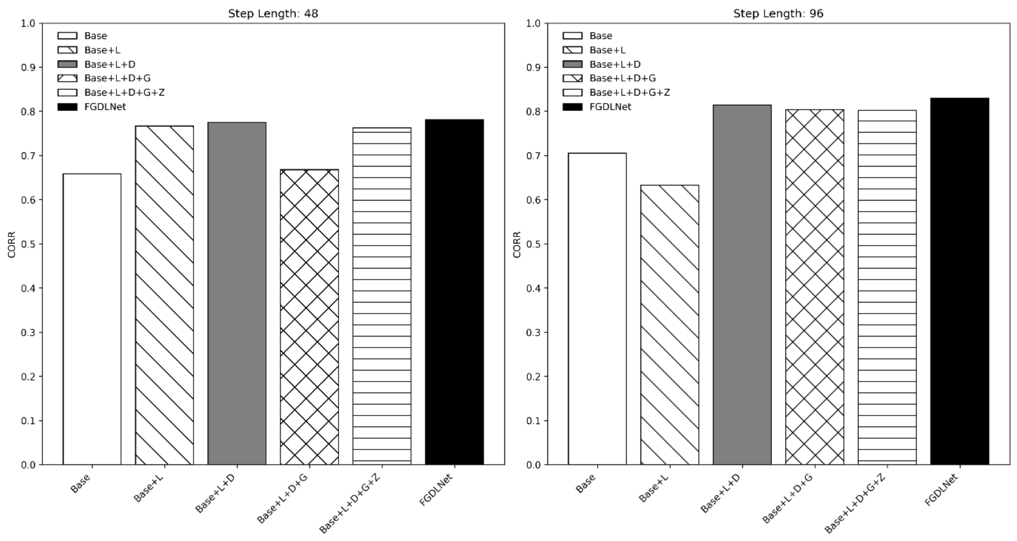

Figure 6: Ablation study

performs well at certain time steps, its consistency and accuracy across the entire time series are inferior to those of FGDLNet.

4.7 Ablation experiments

The FGDLNet model consists of five core modules: linear prediction, single-channel processing, multi-scale feature extraction enhanced with Graph Neural Networks, hybrid attention mechanism, and an improved loss function. To validate the effectiveness of each module in the FGDLNet model, six network variations were designed: (1) Base, which is the basic Transformer model; (2) Base+L, where the Decoder is removed and linear direct prediction is applied; (3) Base+L+D, where single-channel processing is applied for prediction; (4) Base+L+D+G, which incorporates multi-scale feature extraction and the GNN module; (5) Base+L+D+G+Z, which adds the hybrid attention mechanism; (6) FGDLNet, which incorporates the improved loss function. The results of the ablation experiments for prediction lengths of 48 and 96 steps are shown in Figure 6.

4.7.1 Efficiency and running time analysis

In this section, we provide a detailed analysis of the efficiency and runtime of the proposed FGDLNet model, comparing it with other models from the ablation experiments to assess the computational efficiency and prediction accuracy.

We selected three representative models from the ablation experiments for this analysis:

Base: The traditional Transformer model, which serves as the baseline for comparison.

Base+L: This model removes the Decoder and uses linear direct prediction, allowing us to isolate the effect of linear prediction without the full Transformer architecture.

FGDLNet: This model includes all core modules, including single-channel processing, multi-scale feature extraction with GNN, hybrid attention, and the improved loss function.

In this section, we analyze the theoretical time complexity of three models: Base, Base+L, and FGDLNet. The Base model, which corresponds to the original Transformer architecture, has a time complexity of $O(L^2d)$, where L is the length of the input sequence and d is the dimensionality of the hidden layer. This quadratic complexity arises from the self-attention mechanism, which requires pairwise calculations for all tokens in the sequence. The Base+L model, which removes the Decoder and applies linear direct prediction, maintains the same time complexity for the self-attention mechanism as Base, i.e., $O(L^2d)$. However, the linear prediction part has a complexity of $O(Ld)$, which reduces the overall computational burden by eliminating the Decoder. In contrast, FGDLNet, which incorporates local window attention, global linear attention, and a Graph Neural Network for feature extraction, offers a more efficient approach. The time complexity of FGDLNet for the attention mechanism is reduced to $O(wLd)$, where w is the window size, making it more efficient for long sequences compared to the quadratic complexity of the original Transformer. Additionally, the inclusion of GNN for extracting internal features from the single channel increases the time complexity to $O(T^2D)$, where T represents the matrix multiplication for the self-adjacency matrix and D is the feature dimension used in the GNN. This theoretical analysis highlights the computational efficiency of FGDLNet in processing long sequence data, as it reduces the time complexity through hybrid attention mechanisms while adding the computational cost of GNN feature extraction. Table 5 shows the operational efficiency and mean absolute error of three models.

Table 5: Runtime Efficiency and MAE Comparison of Models

model	Training time	MAE
Base	1243.14	0.2948
Base+L	1007.17	0.3677
FGDLNet	1156.47	0.1701

Based on the table data, the Base model has the longest runtime. The Base+L model improves efficiency but with a decrease in accuracy. In contrast, the FGDLNet model achieves the best prediction accuracy while maintaining a good runtime. Therefore, FGDLNet outperforms the other models in terms of both prediction accuracy and efficiency.

5 Discussion

The Transformer model has achieved significant success in traditional time series forecasting, particularly in handling long sequences, where it effectively captures global dependencies. However, the Transformer model suffers from high computational complexity, especially with the self-attention mechanism, which has a time complexity of $O(n^2)$, where nnn is the sequence length. This results in a substantial increase in computational burden when the sequence length is large, particularly in resource-constrained environments.

The DLinear model is based on a simple linear regression approach for time series forecasting, which offers computational efficiency, especially for large-scale data. Its advantage lies in its simplicity and efficiency, making it suitable for scenarios requiring fast computation with relatively simple data patterns. However, its drawback is its limited ability to handle complex patterns and long-term dependencies in time series data, resulting in inferior predictive performance compared to more complex models.

FGDLNet simplifies the model structure by removing the Decoder module from the Transformer. Unlike the traditional Transformer, which relies on the Decoder module for generating the output sequence—requiring substantial computation and complex parameter adjustments—FGDLNet replaces it with a linear layer, directly connecting and predicting the output. This modification reduces the computation process and lowers the time complexity. Additionally, FGDLNet incorporates a Graph Neural Network module, enabling it to enhance dependencies between time steps within a single channel. This improvement allows FGDLNet to capture long-term dependencies and complex patterns more effectively than the traditional Transformer and DLinear. Consequently, FGDLNet enhances model expressiveness and prediction accuracy while maintaining relatively low computational complexity.

6 Conclusion

This paper presents FGDLNet, a time series forecasting model based on the Transformer architecture, designed to overcome the challenges of long sequence prediction. By removing the Decoder module from the traditional Transformer, incorporating multi-scale feature extraction, and employing a hybrid attention mechanism, FGDLNet significantly improves both prediction accuracy and computational efficiency.

FGDLNet addresses the high computational complexity of the traditional Transformer model, which arises from the self-attention mechanism. By using local window attention and global linear attention, FGDLNet reduces time complexity, making it more efficient for long sequence inputs. Additionally, the hybrid attention mechanism captures both local and global dependencies, while the integration of Graph Neural Networks for multi-scale feature extraction enhances the model's ability to capture complex dependencies in time series data.

Experimental results demonstrate that FGDLNet outperforms traditional methods in terms of both prediction accuracy and computational efficiency. The model strikes a good balance between accuracy and efficiency, making it suitable for long sequence forecasting tasks. Future work could focus on further optimizing GNN techniques, integrating multi-modal data, and exploring cross-domain applications to further enhance the model's predictive power and generalization ability.

Conflict of Interest

The authors confirm that the content of this article has no conflict of interest.

Acknowledgement

This research was supported by the Key Scientific and Technological Project of Henan Province (242102210058).

References

[1] Vaswani A. Attention is all you need(2023). Advances in Neural Information Processing Systems, *NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems* Pages 6000 - 6010. https://doi.org/10.48550/arXiv.1706.03762

[2] Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., and Zhang, W. (2021). Informer: Beyond efficient transformer for long sequence time-series forecasting. In Proceedings of the *AAAI conference on artificial intelligence* Vol. 35, No. 12, pp. 11106-11115. https://doi.org/10.1609/aaai.v35i12.17325

[3] Kitaev, Nikita ,Łukasz Kaiser, and Anselm Levskaya. (2020)."Reformer: The efficient transformer." *arXiv preprint* arXiv:2001.04451 . https://doi.org/10.48550/arXiv.2001.04451

[4] Wu, H., Xu, J., Wang, J., & Long, M. (2021). Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems*, Vol.34, 22419-22430. https://doi.org/10.48550/arXiv.2106.13008

[5] Zeng A, Chen M, Zhang L, et al(2023). Are Transformers Effective for Time Series Forecasting? Proceedings of the *AAAI Conference on Artificial Intelligence*. Vol.37(9): 11121-11128. https://doi.org/10.1609/aaai.v37i9.26317

[6] Liu Y, Hu T, Zhang H, et al(2023). iTransformer: Inverted Transformers are Effective for Time Series Forecasting. *arXiv preprint* arXiv:2310.06625, 2023. https://doi.org/10.48550/arXiv.2310.06625

[7] Xu, Zhijian, Ailing Zeng, and Qiang Xu (2023). "FITS: Modeling time series with $10 k $ parameters." .Montavon G, Orr G B, Müller K R. Neural Networks: Tricks of the Trade. *arxiv preprint* arxiv:2307.03756 https://doi.org/10.48550/arXiv.2307.03756

[8] Cao, D., Jia, F., Arik, S. O., Pfister, T., Zheng, Y., Ye, W., & Liu, Y. (2023). Tempo: Prompt-based generative pre-trained transformer for time series forecasting. *arxiv preprint* arxiv:2310.04948. https://doi.org/10.48550/arXiv.2310.04948

[9] Yitian Zhang, Liheng Ma, Soumyasundar Pal, Yingxue Zhang, Mark Coates (2023). Multi-resolution Time-Series Transformer for Long-term Forecasting. *arxiv preprint* arXiv:2311.04147v2 https://doi.org/10.48550/arXiv.2311.04147

[10] Arjun Ashok, Étienne Marcotte, Valentina Zantedeschi, Nicolas Chapados, Alexandre Drouin (2023). TACTiS-2: Better, Faster, Simpler Attentional Copulas for Multivariate Time Series.*arxiv preprint* arxiv:2310.01327. https://doi.org/10.48550/arXiv.2310.01327

[11] Yang, Z., Liu, L., Li, N., & Tian, J. (2022). Time series forecasting of motor bearing vibration based on informer. *Sensors*, 22(15), 5858. https://doi.org/10.3390/s22155858

[12] Gong, M., Zhao, Y., Sun, J., Han, C., Sun, G., & Yan, B. (2022). Load forecasting of district heating system based on Informer. *Energy*, 253, 124179. https://doi.org/10.1016/j.energy.2022.124179

[13] Pan, K., Lu, J., Li, J., & Xu, Z. (2023). A Hybrid Autoformer Network for Air Pollution Forecasting Based on External Factor Optimization. *Atmosphere*, 14(5), 869. https://doi.org/10.3390/atmos14050869

[14] Feng, H., & Zhang, X. (2023). A novel encoder-decoder model based on Autoformer for air quality index prediction. *Plos one*, 18(4), e0284293. https://doi.org/10.1371/journal.pone.0284293

[15] Huang, Y., & Wu, Y. (2023). Short-Term Photovoltaic Power Forecasting Based on a Novel Autoformer Model. *Symmetry*, 15(1), 238. https://doi.org/10.3390/sym15010238

[16] Ahmed N K, Atiya A F, Gayar N E, et al. An empirical comparison of machine learning models for time series forecasting[J]. Econometric reviews, 2010, 29(5-6): 594-621. https://doi.org/10.1080/07474938.2010.481556

[17] Frissou N, Kimour M T, Selmane S. Optimized Support Vector Regression for Predicting Leishmaniasis Incidences[J]. *Informatica (Slovenia)*, 2021, 45(7). https://doi.org/10.31449/inf.v45i7.3665

[18] Liu Y, Pan B. Profit estimation model and financial risk prediction combining multi-scale convolutional feature extractor and BGRU model[J]. *Informatica (Slovenia)*, 2024, 48(11). https://doi.org/10.31449/inf.v48i11.5941

[19] Wang T, Yu J, Cao Y, et al. Fault Prediction of CNC Machine Tools Based on Gutenberg-Richter Law and Fuzzy Neural Networks[J]. *Informatica (Slovenia)*, 2024, 48(18). https://doi.org/10.31449/inf.v48i18.6292

[20] Flores A, Tito-Chura H. Multi-Step Forecasting of Guillain Barré Cases using Deep Learning[J]. *Informatica (Slovenia)*, 2024, 48(20). https://doi.org/10.31449/inf.v48i20.6358