Supplementary Material for CrossGNN: Confronting Noisy Multivariate Time Series Via Cross Interaction Refinement

Anonymous Author(s)
Affiliation
Address
email

Appendix

The supplementary material consists of:
A. Experimental Details: Descriptions of datasets, baseline methods, and implementation details.
B. Additional Experimental Results: Robustness analysis on noisy data, sensitivity analysis on hyperparameters, and visualization of forecasting results.
C. Derivation of Computational Complexity: A theoretical proof of the linear computation complexity of CrossGNN.

A Experimental Details

A.1 Datasets

We conduct extensive experiments on 8 real-world datasets following [4]. The interval length, time step number, and the variable number of each real-world dataset are presented in Table 1. The detailed dataset descriptions are as follows:
1) ETTh (ETTh1, ETTh2, ETTm1, ETTm2) consists of two hourly-level datasets (ETTh) and two 15minute-level datasets (ETTm). Each of them contains seven oil and load features of electricity transformers from July 2016 to July 2018.
2) Weather includes 21 indicators of weather, such as air temperature, and humidity. Its data is recorded every 10 min for 2020 in Germany.
3) Traffic describes hourly road occupancy rates measured by 862 sensors on San Francisco Bay area freeways from 2015 to 2016.
4) Exchange-rate collects the daily exchange rates of 8 countries from 1990 to 2016.
5) Electricity contains hourly electricity consumption (in Kwh) of 321 clients from 2012 to 2014.

Table 1: The statistics of the datasets for MTS forecasting.

Datasets	ETTh1	ETTh2	ETTm1	ETTm2	Weather	Traffic	Exchange-rate	Electricity
Interval Length	1 Hour	1 Hour	15 Minutes	15 Minutes	10 Minutes	1 Hour	1 Day	1 Hour
Time step #	17,420	17,420	69,680	69,680	52,696	17,544	7,588	26,304
Variable #	7	7	7	7	21	862	8	321

A.2 Baseline Methods

We briefly describe the selected 7 state-of-the-art baselines as follows:
1) TimesNet [3] is a task-general foundational model for time series analysis that utilizes a modular architecture to unravel intricate temporal variations. A parameter-efficient inception block is leveraged

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

to capture intra-period and inter-period variations in 2D space.
2) ETSformer [2] is a Transformer-based model for time-series forecasting that incorporates inductive biases of time-series structures and introduces novel exponential smoothing attention (ESA) and frequency attention (FA) to improve performance.
3) DLinear [6] is a simple linear-based model combined with a decomposition scheme.
4) FEDformer [7] is a Transformer-based model that uses the seasonal-trend decomposition with frequency-enhanced blocks to capture cross-time dependency for forecasting.
5) Autoformer [4] is a Transformer-based model using decomposition architecture with an auto-Correlation mechanism to capture cross-time dependency for forecasting.
6) Pyraformer [1] is a Transformer-based model learning multi-resolution representation of the time series by the pyramidal attention module to capture cross-time dependency for forecasting.
7) MTGNN [5] explicitly utilizes cross-variable dependency using GNN. A graph learning layer learns a graph structure where each node represents one variable in MTS.

A.3 Implementation Details

To ensure a fair comparison, the look-back window size is set to 96, which is consistent with all baselines. We set the scale numbers S to 5 and set K to 15 for all datasets, as sensitivity experiments have shown that S does not have a significant impact beyond 5 and CrossGNN is not sensitive to K. Besides, the dimension of the channel is set to 16 based on efficiency considerations. Additionally, the mean squared error (MSE) is used as the loss function. For the learning rate, a grid search is conducted among [5e-3, 1e-3, 5e-4, 1e-4, 5e-5, 1e-5] to obtain the most suitable learning rate for all datasets. The total number of training epochs is set to 10, and training would be terminated early if the validation loss does not decrease for three consecutive rounds. The model is implemented in PyTorch 1.8.0 and trained on a single NVIDIA Tesla V100 PCIe GPU with 16GB memory.

B Additional Experimental Results

B.1 Analysis on Robustness Against Noise

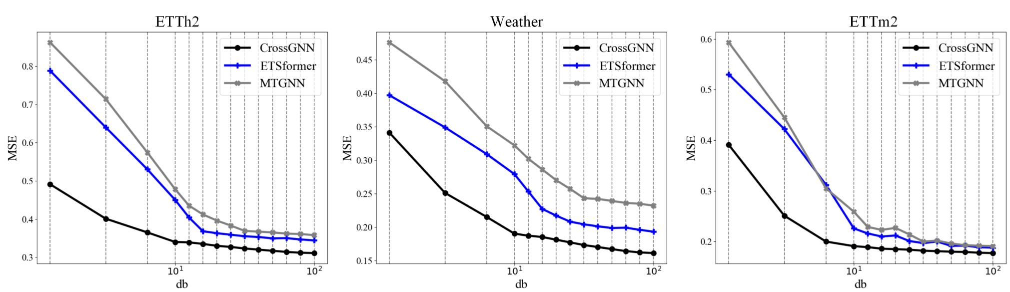

Figure 1: The MSE results (Y-axis) of models on ETTh2, ETTm2 and Weather with different signal-to-noise ratio (SNR).

To evaluate the robustness of CrossGNN against noise, we add different intensities of Gaussian white noise to the original MTS and observe the performance changes. As the intensity of Gaussian white noise increases, the signal-to-noise ratio (SNR) gradually decreases from 100 dB to 0 dB. Figure 1 shows the MSE results of CrossGNN, ETSformer [2] and MTGNN [5] on ETTh2, ETTTm2, and Weather under different SNR. As the SNR decreases from 100db to 0db, the mean square error (MSE) of CrossGNN increases more slowly than MTGNN and ETSformer.

Taking the results on the ETTm2 dataset as an example, when the noise intensity increases at the beginning (i.e., SNR decreases from 100db to 10db), the prediction accuracy of MTGNN and ETSformer becomes unstable. Their prediction accuracy drops more rapidly when the noise intensity suddenly increases (i.e., SNR decreases from 10db to 0db). In contrast, CrossGNN maintains overall stability, and its performance degrades more slowly. The quantitative results demonstrate that CrossGNN exhibits good robustness against noisy data and has a great advantage when dealing with unexpected fluctuations. Such improvements benefit from the explicit modeling of respective cross-scale and cross-variable interactions.

B.2 Sensitivity Analysis

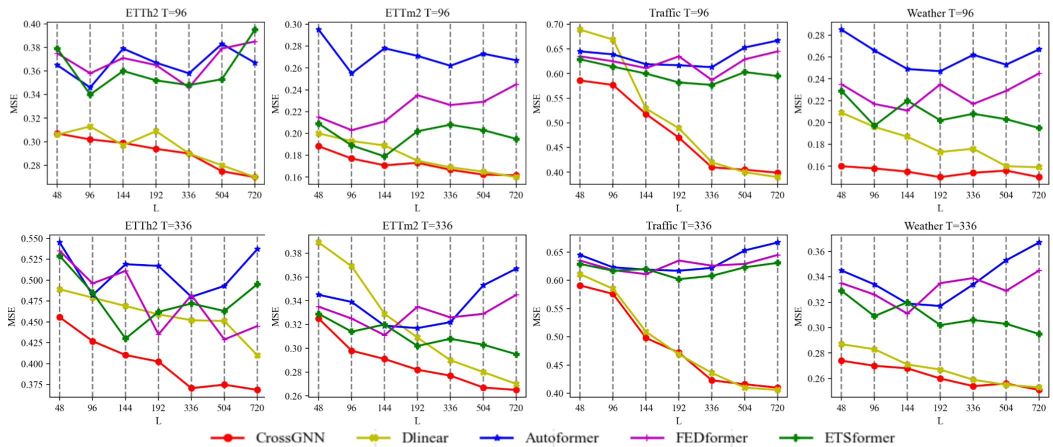

Figure 2: The MSE results (Y-axis) of models with different look-back window sizes (X-axis) on ETTh2, ETTm2, Traffic and Weather. The first row shows the performance when the prediction horizon is 96, while the second row shows the performance when the prediction horizon is 336.

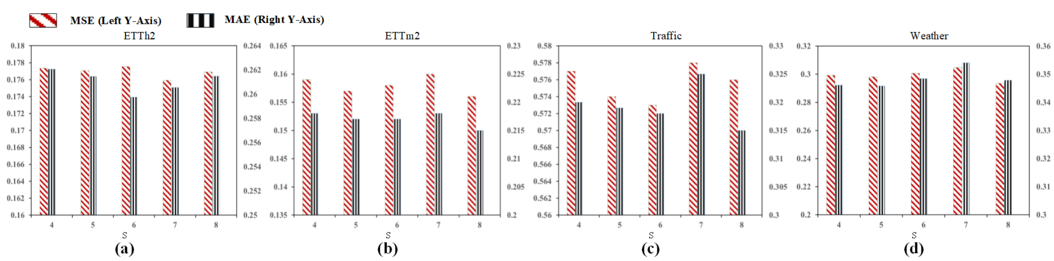

Figure 3: The MSE (left Y-axis) and MAE results (right Y-axis) of CrossGNN with different number of scales (X-axis) on ETTh2, ETTm2, Traffic, and Weather.

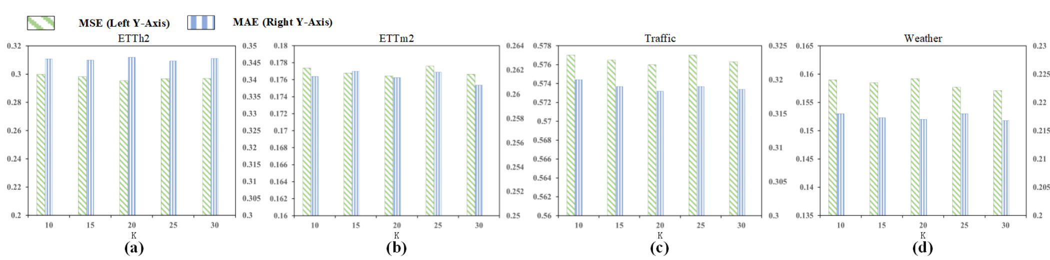

Figure 4: The MSE (left Y-axis) and MAE results (right Y-axis) of CrossGNN with different K (X-axis) on ETTh2, ETTm2, Traffic, and Weather.

Look-Back Window Size Figure 2 shows the MSE results of models with different look-back window sizes on four datasets. As the window size increases, the performance of Transformer-based models fluctuates while CrossGNN constantly improves. This indicates that the attention mechanism of Transformer-based models may focus much more on the temporal noise, while CrossGNN can better extract the relationships between different time nodes via the Cross-Scale module.

Number of Scales We vary the number of scales from 4 to 8 and report the MSE and MAE results on ETTh2, ETTm2, Traffic, and Weather. As shown in Figure 3, We observe that the performance improvement becomes less significant after a certain number of scales (i.e., 5), indicating that a certain scale size is sufficient to eliminate most of the effects of temporal noise.

Number of Temporal Node Neighbors The number of temporal neighboring nodes is primarily determined by the hyperparameter K. As depicted in Figure 4, we conducted experiments with different K values, including 10, 15, 20, 25, and 30, and observed that CrossGNN is not sensitive to the number of K. This suggests that effective cross-scale interaction can be achieved by focusing only on strongly correlated time nodes.

B.3 Visualization of Forecasting Results of Different Models

We present the visualization of forecasting results of CrossGNN and other baseline models on 8 datasets in Figure 5 and Figure 6. These datasets exhibit diverse temporal patterns, with 96-steps input length and output horizon. It can be observed that the prediction results of the Transformer-based model are significantly affected by noise, resulting in fluctuations. In contrast, the prediction results of CrossGNN are less affected by noise, and the predicted values are closer to the true results.

For example, considering the forecasting results on the Traffic dataset, there are three unexpected noise points (i.e., irregularly high points) in the input data. During prediction, the attention mechanism of the Transformer-based model may focus on the noisy points, leading to a bias towards higher output predictions. As a result, although the Transformer-based model seems to capture the periods of the time series, it fails to produce accurate predictions. In contrast, CrossGNN is unaffected by these three noisy data points and generates predictions that are closer to the ground truth. While Transformer-based models struggle to capture the scale and bias of future data due to unexpected noise in the input data, CrossGNN outperforms other models in terms of both scale and bias in forecasting.

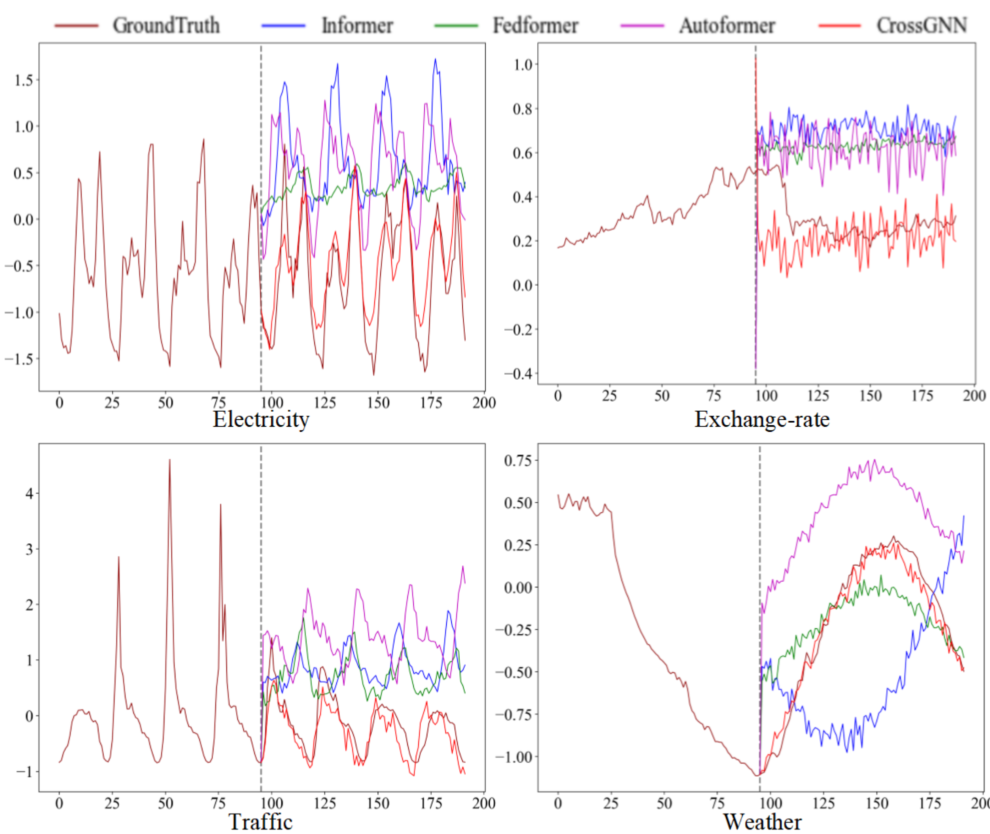

Figure 5: Visualization of 96-step forecasting results on Electricity, Exchange-rate, Traffic, and Weather, and the look-back window size is set as 96.

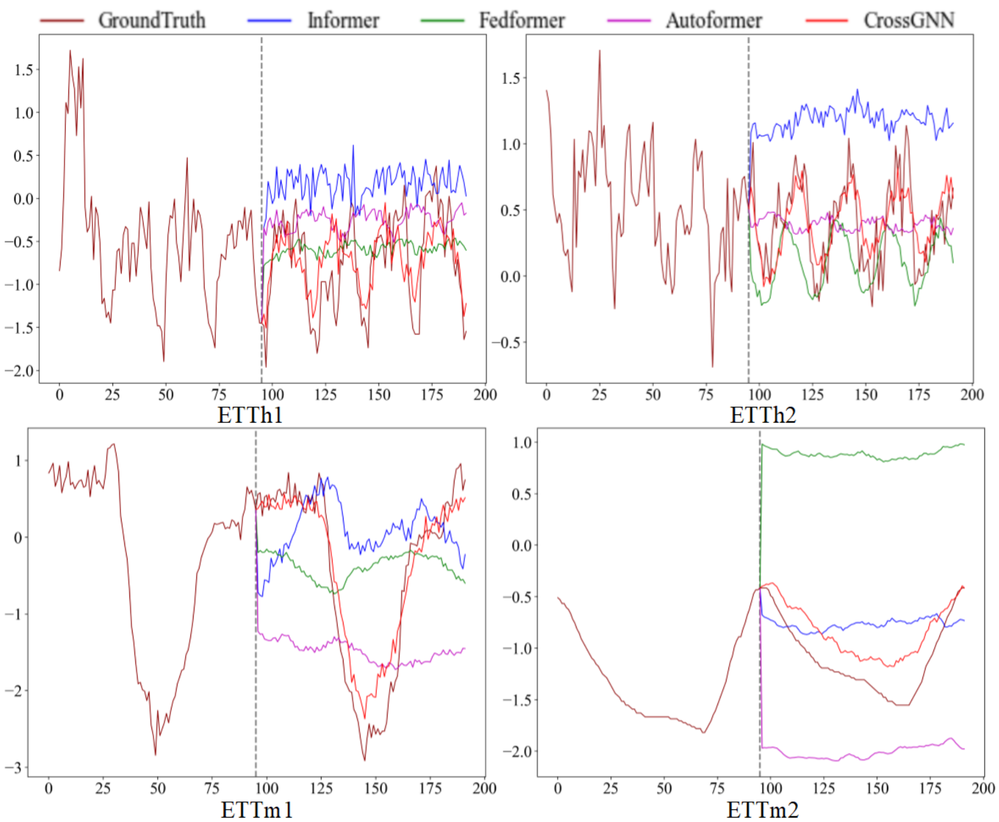

Figure 6: Visualization of 96-step forecasting results on ETTh1, ETTh2, ETTm1 and ETTm2, and the look-back window size is set as 96.

C Derivation of Computational Complexity

In this section, we theoretically prove that the time and space complexity of the Cross-Scale module and Cross-Variable module in CrossGNN are both linear. We have organized the notations used in Table 2 for ease of reading.

Table 2: Meaning of notations

Notation	Meaning
v_i	The i-th time node
S	Number of scales
s	Index of the scale
p_s	Period length of the s-th scale
L	Original input length (i.e., look-back window size)
$L(s)$	Time length at the s-th scale
L'	Total length of concatenated multi-scale time series
K^{scale}	The hyperparameter to control temporal neighbor numbers
K^{var}	The hyperparameter to control variable neighbor numbers
k_s	The temporal neighbor numbers at the s-th scale
$A(v_i)$	Total temporal neighbor node number correlated to v_i
A	Total correlated temporal node pair number
D	Variable numbers

Proposition 1. *The time and space complexity for the Cross-scale GNN is $O(K^{scale} \times \ln S \times L)$ and amounts to $O(L)$ when S and K^{scale} are constants w.r.t. L.*

Proof. To improve readability, we substitute K for K^{scale}. Denote $L(s)$ as the number of time nodes at s-th scale:

$$L(s) = \lfloor \frac{L}{p_s} \rfloor, 1 \leq s \leq S, \tag{1}$$

where p_s is the corresponding period length of the s-th scale and L is the original input length (i.e., look-back window size). L' is the sum of time nodes at different scales, and it could be expressed by:

$$L' = \sum_{s=1}^{S} L(s) = \sum_{s=1}^{S} \lfloor \frac{L}{p_s} \rfloor \leq \sum_{s=1}^{S} \lfloor \frac{L}{s} \rfloor \leq L \sum_{s=1}^{S} \frac{1}{s} \approx L(\ln S + \epsilon + \frac{1}{2S}), \tag{2}$$

where $lnS + \epsilon + \frac{1}{2S} \approx \sum_{s=1}^{S} \frac{1}{S}$ is the approximate summation formula for the harmonic series, and ϵ is the Euler-Mascheroni constant.

For a time node, we set its scale-sensitive time node neighbor numbers to $k_s = \lceil \frac{K}{p_s} \rceil$ at s-th scale. Since the trend-aware neighbor nodes are defined as its previous node and next node at the current scale, the number of trend-aware neighbor nodes can reach 2 when these nodes do not overlap with the scale-sensitive neighbor nodes. However, when there is overlap, the number of trend-aware neighbor nodes can be 0 or 1.Therefore, the maximum neighbor node number of v_i is given by:

$$\begin{aligned} A(v_i) &\leq \sum_{s=1}^{S} k_s + 2 && (3)\\ &= K \sum_{s=1}^{s=S} \frac{1}{p(s)} + 2 && (4)\\ &\leq K(\frac{1}{1} + \frac{1}{2} + ... + \frac{1}{S}) + 2 && (5)\\ &\approx K(\ln S + \epsilon + \frac{1}{2S}) + 2, && (6) \end{aligned}$$

Total correlated node pair number is expressed as:

$$A = L' \times A(v_i) \leq L(\ln S + \epsilon + \frac{1}{2S})(K(\ln S + \epsilon + \frac{1}{2S}) + 2) \approx 2K \times \ln(S) \times L. \tag{7}$$

Consequently, the complexity of the proposed cross-scale GNN is:

$$O(A) \leq O(2K \times \ln(S) \times L). \tag{8}$$

Since K and S are all constant terms that are independent of the length L and remain fixed when L changes, the complexity can be further reduced to $O(L)$. $\square$

Proposition 2. *The time and space complexity for the Cross-variable GNN is $O(K^{var} \times D)$ and amounts to $O(D)$ when K^{var} is a constant w.r.t. D.*

Proof. Without loss of generality, we assume that the number of homogeneous and heterogeneous correlated nodes for each variable are both K^{var}. For a cross-variable graph, there are a total of $\sum_{i=1}^{D} 2K^{var} = 2K^{var}D$ correlated variable node pairs. Correspondingly, since the complexity of graph computation is related to the number of edges, the time and space complexity of cross-variable GNN are both $O(K^{var} \times D)$. As K^{var} is a constant that is independent of D, its complexity is linear (i.e., $O(D)$). $\square$

References

[1] Shizhan Liu, Hang Yu, Cong Liao, Jianguo Li, Weiyao Lin, Alex X Liu, and Schahram Dustdar. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *International conference on learning representations*.

[2] Gerald Woo, Chenghao Liu, Doyen Sahoo, Akshat Kumar, and Steven C. H. Hoi. Etsformer: Exponential smoothing transformers for time-series forecasting. 2022.

[3] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. In *International Conference on Learning Representations*, 2023.

[4] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems*, 34:22419–22430, 2021.

[5] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020.

[6] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? *arXiv preprint arXiv:2205.13504*, 2022.

[7] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *Proc. 39th International Conference on Machine Learning (ICML 2022)*, 2022.