WAG-NAT: Window Attention and Generator Based Non-Autoregressive Transformer for Time Series Forecasting

Yibin Chen[1], Yawen Li[2], Ailan Xu[3], Qiang Sun[1(✉)], Xiaomin Chen[1], and Chen Xu[1]

[1] School of Information Science and Technology, Nantong University, Nantong 226019, China
1910110082@stmail.ntu.edu.cn, {sunqiang,chenxm,xuchen}@ntu.edu.cn
[2] Shanghai Academy of Environmental Sciences, Shanghai 200235, China
[3] Jiangsu Province Nantong Environmental Monitoring Centre, Nantong 226019, China

Abstract. Time series forecasting plays a crucial part in many real-world applications. Recent studies have proven the power of Transformer to model long-range dependency for time series forecasting tasks. Nevertheless, the quadratic computational complexity of self-attention is the major obstacle to application. Previous studies focus on structural adjustments of the attention mechanism to achieve more efficient computation. In contrast, local attention has better performance than full attention in feature extraction and computation simplification due to the sparsity of the attention mechanism. Besides, in practice, the speed of inference is more significant, which is also a key factor. In response to these, we develop a novel non-autoregressive Transformer model based on window attention and generator, namely **WAG-NAT**. The generator allows one-step-forward inference. The window attention module contains a window self-attention layer to capture local patterns and a window interaction layer to fuse information among different windows. Experimental results show that WAG-NAT has a distinct improvement in prediction accuracy compared with RNNs, CNNs, and other previous Transformer-based models across various benchmarks. Our implementation is available at https://github.com/cybisolated/WAG-NAT.

Keywords: Time series forecasting · Non-autoregressive Transformer · Window attention · Deep learning

1 Introduction

By conveniently predicting future events or indicators, deep learning-based time series forecasting effectively improves the quality of decision-making. Therefore, time series forecasting plays a significant role in many fields, such as weather forecasting [6,9,17], logistics management [18], and economics [16].

L. Iliadis et al. (Eds.): ICANN 2023, LNCS 14259, pp. 293–304, 2023.
https://doi.org/10.1007/978-3-031-44223-0_24

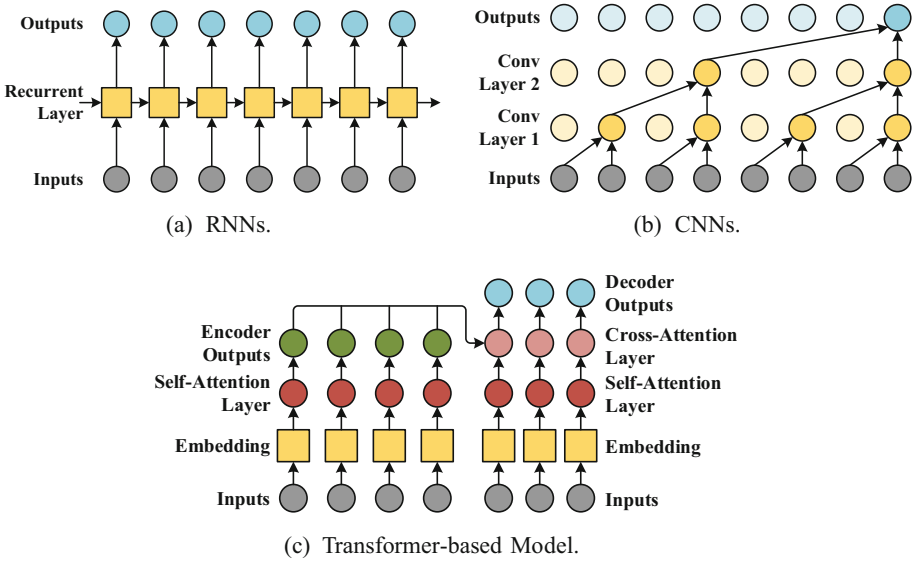


Fig. 1. Three typical models for time series forecasting.

Figure 1 shows three mainstream deep learning models for time series forecasting based on sequence modeling as follows: (1) recurrent neural networks (RNNs), (2) convolutional neural networks (CNNs), and (3) Transformer-based networks.

RNNs are widely applied for time series forecasting and are developed into new variants, e.g., LSTM and GRU. RNNs can model complex temporal relationships within sequential data, therefore taking advantages in occasions emphasizing the importance of time order. Nevertheless, it is challenging for RNNs to train deep networks and model long-range dependency due to the vanishing gradient problem [15]. Moreover, RNNs' weakness of modeling complicated dynamic patterns could result in the underfitting of intricate data.

CNNs have been applied to the task of time series forecasting because of their ability to extract features from sequential data [12]. CNNs capture local patterns and extract relevant features from time series data. These characteristics elevate the suitability of CNNs in the local-pattern-dominated data processing. The emergency of temporal convolutional network (TCN) [4] is one of the milestones facilitating CNNs to shine in time series forecasting. TCN couples residual connections and dilated causal convolutions to model time series data, thus achieving significant results in various tasks. However, CNNs' capture of long-term dependency is not as effective as other models, e.g., LSTM. As a result, the prediction accuracy is relatively low for data with long-term dependency.

Transformer was originally developed for natural language processing (NLP), and now it has become the most popular and powerful tool for time series forecasting. A few fundamental studies have applied time series forecasting with

Transformer-based models. [13] proposed the first Transformer model with only a decoder for time series forecasting, which introduces causal convolution to produce queries and keys before the attention layer. The model incorporates local and global patterns, and outperforms the RNN-based models across various benchmarks. [19] has developed a novel paradigm that employs a vanilla Transformer to predict time series data. The historical sequential data is utilized as the input for the encoder, while training and inference procedures are similar to that of NLP. However, this approach preserves the autoregressive decoder architecture in NLP, resulting in large cumulative errors especially when predicting long sequences. The work in [20] extends the start token technique of NLP's "dynamic decoding" into a generative way. This generates an output sequence of a specified length and solves the problem of cumulative errors to a certain extent. Besides, the Informer model [20] has effectively reduced the computational complexity of the self-attention mechanism and has broken the memory bottleneck in stacking layers for long inputs. However, the start token technique requires a shift of partial encoder inputs to the decoder and filling the prediction position with zeros. This sacrifices the prediction capacity of the decoder and the padded zeros contribute little to prediction accuracy.

In this work, to overcome the challenges mentioned above, we build a novel generator between the encoder and decoder inspired by the work of [5,8]. This generator incorporates an embedded encoder sequence and a learnable positional encoding vector to create the required input sequences, therefore predict sequences without the restriction of the length. In addition, under the precondition that temporal effects might not correspond significantly with the increasing time intervals [1], a particular point in a time series does not attend to another point that is far away from it. We do not think it is necessary to perform attention computation to the whole sequence due to the sparsity of attention weights [13,20]. Therefore, we adopt the window attention mechanism [14] and use group convolution [10] to merge the information from different windows. The experimental result shows that the computational complexity of attention is effectively reduced. To sum up, our main contribution are as follows:

1. We design a novel non-autoregressive Transformer-based architecture for time series forecasting, including an encoder, a generator, and a decoder. The model generates the predicted sequences of specific length within one step forward regardless of training or inference.
2. We refine the structure of the generator to effectively combine location and future-known information, thus generating optimal sequences.
3. We introduce the window attention technique with complexity $\mathcal{O}(L)$ to enhance locality for time series forecasting. Different windows interact through group convolution to fully integrate information.

2 Problem Definition

We take multi-step forecasting as a scenario, and suppose y_t^i is the ground truth value of entity i at time t. Each entity denotes a logical grouping, such

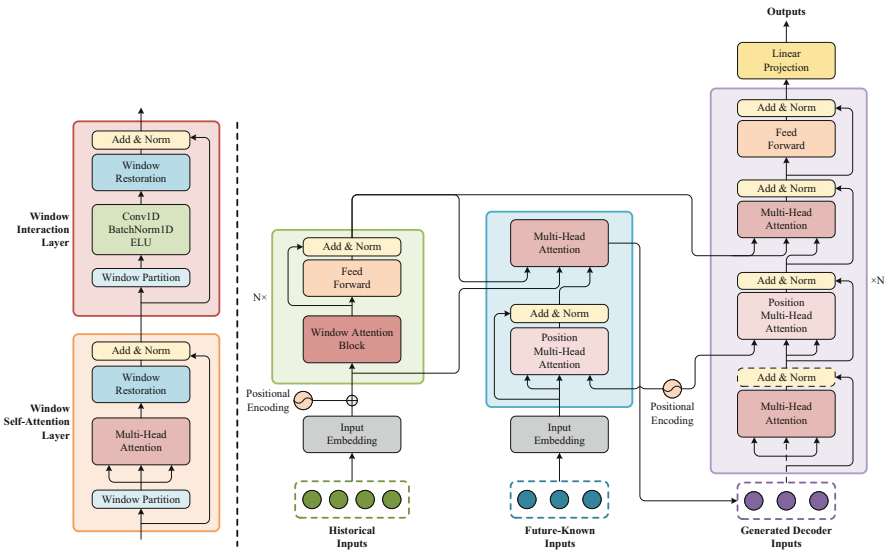
as measurements from items for sale, or data from different weather stations. For given observations of target $y^i_{t-k:t} = \{y^i_{t-k}, \cdots, y^i_t\}$ and covariates $\boldsymbol{x}^i_{t-k:t} = \{\boldsymbol{x}^i_{t-k}, \cdots, \boldsymbol{x}^i_t\}$ over a look-back window of size k, future known exogenous inputs $\boldsymbol{z}^i_{t-k:t+\tau} = \{\boldsymbol{z}^i_{t-k}, \cdots, \boldsymbol{z}^i_{t+\tau}\}$ and static metadata $\boldsymbol{s}^i_{t-k:t+\tau} = \{\boldsymbol{s}^i_{t-k}, \cdots, \boldsymbol{s}^i_{t+\tau}\}$ associated with the entity (e.g., meteorological station number) over the mentioned look-back window and forecasting horizon of size τ, time series forecasting model predicts future values of τ steps $\hat{y}^i_{t+1:t+\tau} = \{\hat{y}^i_{t+1}, \cdots, \hat{y}^i_{t+\tau}\}$, which can be formulated as

$$\hat{y}^i_{t+1:t+\tau} = f(y^i_{t-k:t}, \boldsymbol{x}^i_{t-k:t}, \boldsymbol{z}^i_{t-k:t+\tau}, \boldsymbol{s}^i_{t-k:t+\tau}), \tag{1}$$

where $f(\cdot)$ represents the learnt prediction function by the model. We note $\boldsymbol{x}^i_t, \boldsymbol{z}^i_t$ and $\boldsymbol{s}^i_t$ as a set of variables, and these variables are either real or categorical. To be simplified, the entity index i is omitted from formulas below unless necessary.

3 Proposed Model

In this section, we discuss the proposed WAG-NAT model in detail. WAG-NAT is a Transformer-based model, which adopts an encoder-decoder architecture. As shown in Fig. 2(b), we improve the canonical Transformer architecture on the encoder and the decoder to realize WAG-NAT. Meanwhile, a plug-and-play generator is introduced to produce decoder inputs, which allows the generation of long predicted sequences within one step forward.



(a) Window Attention Block. (b) Architecture of WAG-NAT.

Fig. 2. (a) The detailed architecture of a window attention block; (b) the architecture of WAG-NAT.

3.1 Encoder: Reinforcing Computing Power with Window Attention Mechanism

The encoder is used to extract the long-range dependency, similar to its extraction of semantic information from source language text in translation tasks. The encoder accepts the historical sequence as inputs, i.e., the look-back window. We denote $\boldsymbol{X}_{t-k:t} = \{y_{t-k:t} \circ \boldsymbol{x}_{t-k:t} \circ \boldsymbol{z}_{t-k:t} \circ \boldsymbol{s}_{t-k:t}\}$ as the encoder inputs, where $\circ$ represents the concatenation operation across feature dimension.

Feature Representation. Regarding feature representations of the Transformer-based models, most only allow input of real variables, therefore categorical variables are not utilized. However, those categorical variables could be crucial to the prediction (e.g., wind direction for weather forecasting). Here we apply entity embeddings [7] as feature representations for categorical variables, and add those to the embedded real variables. Besides, the sinusoidal positional encoding technique is used, then we get the representation of $\boldsymbol{X}_{t-k:t}$, denoted by $\mathcal{X}_{t-k:t} \in \mathbb{R}^{k \times d_m}$, where d_m is the embedding dimension.

Window Attention Block. A canonical self-attention module multiplies the input sequence with three different weight tensors to generate queries, keys and values, denoted by Q, K and V. Then, the scaled dot-product is performed, and can be formulated as

$$\mathcal{A}(Q, K, V) = \frac{\text{Softmax}(QK^\top)}{\sqrt{d_m}} V \tag{2}$$

According to the above formula, each query attends to all the keys, which causes quadratic time complexity over the length of queries. Generally speaking, if the selected lagged features have a large distance to the predicted point, these features may not contribute effectively to the accuracy of prediction. Therefore, we propose a *window attention block*, as presented in Fig. 2(a). Specifically, we divide the input sequence into multiple windows equidistantly. The window self-attention layer performs a self-attention operation within each window, and each query only attends to the keys from the same window. After that, various windows interact through the window interaction layer, where the information from different windows is fully integrated.

Window Self-Attention Layer. Denoting L the length of input sequence and w the length of input sequence in each window, called by window size, we attain $M = L/w$ windows. Then we get queries, keys and values of each window, thus obtaining M output sequences $\omega_i = \mathcal{A}(Q_i, K_i, V_i)$, where $i = 1, \cdots, M$ and $Q_i, K_i, V_i \in \mathbb{R}^{w \times d_m}$. The computational complexity of a canonical multi-head self-attention (MSA) module is $\mathcal{O}(\text{MSA}) = 4Ld_m^2 + 2L^2 d_m$, and that for a window multi-head self-attention (WMSA) module is

$$\begin{aligned} \mathcal{O}(\text{WMSA}) &= M(4wd_m^2 + 2w^2 d_m) \\ &= 4Ld_m^2 + 2wLd_m, \end{aligned} \tag{3}$$

where the latter term is linear to L when window size w is fixed. That is to say, the length of sequences is the bottleneck for a MSA module, but are totally affordable for a WMSA module.

Window Interaction Layer. Local information is sufficiently extracted within each window through the window self-attention layer. However, window self-attention lacks connections across windows and limits itself to local modeling. The group convolution technique is introduced to interact across windows and keep a temporal order, without affecting the efficient computing capacity. Supposing the outputs of window self-attention layer $\boldsymbol{\Omega} = \{\omega_i, \cdots, \omega_M\} \in \mathbb{R}^{M \times w \times d_m}$ and stacking features for each ω_i, a new set of windows $\boldsymbol{\mathcal{W}} \in \mathbb{R}^{(w \times d_m) \times M}$ can be obtained. Then, features are divided into w groups with respect to channels, and 1-D convolution is performed over each group, which achieves full interaction among windows and efficient computation at the same time.

Finally, a fully-connected layer is applied to produce the final outputs. We note that encoder layers can be stacked, as illustrated in Fig. 2(b), to fulfill further feature extraction depending on the forecasting task.

3.2 Generator: Producing Sequential Outputs Full of Fused Information with Position and Future-Known Covariates

A vanilla Transformer model predicts $\hat{y}_{t+1:t+\tau}$ in an autoregressive manner when inferring [19]. That is to say, τ-step forward is required, leading to large cumulative error and exorbitant cost of computation resources when the forecasting horizon increases. Since the forecasting horizon could be known in advance, we generate the inputs of the decoder within one step forward, and make the parallel inference feasible. The generator accepts future-known features $\boldsymbol{G}_{t+1:t+\tau} = \{\boldsymbol{x}_{t+1:t+\tau} \circ \boldsymbol{z}_{t+1:t+\tau} \circ \boldsymbol{s}_{t+1:t+\tau}\}$ as inputs. Adopting the same embedding strategy as the encoder, we attain the embedded sequences, denoted by $\boldsymbol{\xi}_{t+1:t+\tau} \in \mathbb{R}^{\tau \times d_m}$. In addition, we bring in a position multi-head attention (PMA) module, exactly operating as a canonical attention module despite of adopting positional encoding tensors as queries. Therefore, the positional information is successfully incorporated into the attention procedure. Plus, attended keys and values are position-sensitive, so more temporal information is preserved. Next, a multi-head attention is performed. In this case, queries, keys and values are outputs of the PMA module, outputs of the encoder, and embedded inputs of the encoder, respectively. To explain that, we aim to use sufficiently fused features with position and covariate information (corresponding to the queries) to query the appropriate encoder input features (corresponding to the values) to serve as the input for the decoder. Besides, after the adequate encoding procedure, the encoder output has integrated sufficient local and global information, and could be used as keys to improve query quality (corresponding to the keys). Subsequently, the decoder inputs $\boldsymbol{\phi}_{t+1:t+\tau} \in \mathbb{R}^{\tau \times d_m}$ are generated, which can be directly fed to the decoder without embedding.

3.3 Decoder: Generating Predictions Through One-Step Forward Process

The generated sequence $\phi_{t+1:t+\tau}$ is actually the weighted-sum representation of encoder inputs over length dimension with sufficient feature selection and information fusion. Also, the sequence is the most suitable decoder input. Consequently, the embedding layer is eliminated in the decoder. Unlike the decoder in a vanilla Transformer, WAG-NAT introduces a PMA module between two attention modules, as demonstrated in Fig. 2(b). The PMA module inherits the operation in the generator. With all the required sequential inputs, prediction sequence of length τ can be obtained through a one-step forward procedure, which is impressively time-saving compared with the autoregressive model. Note that the decoder layers are also the same stackable as encoder layers.

4 Experiment

4.1 Datasets

We conduct experiments on several datasets, namely *Electricity Transformer Temperature* [20] (namely {ETTh1, ETTh2} and ETTm1)[1] and *Beijing Multi-Site Air-Quality* (BMSAQ)[2]. The BMSAQ dataset includes hourly air pollutants data from 12 nationally controlled air quality monitoring sites in Beijing, ranging from 1 March 2013 to 28 February 2017. We take "ozone concentration" as the predictive target and the rest as covariates. A brief description of these datasets is listed in Table 1.

4.2 Experimental Settings

Baselines. We select five deep learning-based time series forecasting methods as baselines and divided them into four groups, namely (1) RNNs: {LSTMa [3]}, (2) CNNs: {TCN [4]}, (3) autoregressive Transformers: {vanilla Transformer [19]} and (4) non-autoregressive Transformers: {LogTrans [13], Informer [20]}.

Table 1. A brief depiction of datasets.

Datasets	Number of Variants	Number of Samples	Granularity	Task Type	Data Split (Training/Validation/Testing)
{ETTh1, ETTh2}	7	17,420	1 h	Multivariate	12/4/4 months[a]
ETTm1	7	69,680	15 min		
BMSAQ	13	420,768	1 h	Univariate	38/5/5 months

[a] We follow the data split method of [20].

[1] The ETT dataset was collected from https://github.com/zhouhaoyi/ETDataset.

[2] We obtained the BMSAQ dataset from https://archive.ics.uci.edu/ml/datasets/Beijing+Multi-Site+Air-Quality+Data.

Metrics. For all the benchmarks, two metrics including MSE $= \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ and MAE $= \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$ are adopted to evaluate the performance of models over the test set, where n is the size of prediction window, y_i and $\hat{y}_i$ are the actual and estimated values of predictive target respectively. For multivariate forecasting task, we take the average over different targets.

Training and Tuning Details. We conduct training, testing and tuning procedures on a 12 GB Nvidia RTX 3060 GPU. The normalization technique is adopted to simplify model convergence. For the ETT dataset, we apply z-score normalization, consistent with [20]. For the BMSAQ dataset, we apply min-max normalization and label encoding for both real and categorical inputs respectively to each monitoring site, which is recognized as an independent entity.

Table 2. Ranges of hyperparameters for WAG-NAT on all the datasets.

Datasets	{ETTh1, ETTh2}	BMSAQ	ETTm1
Encoder Length (k)	{24, 48,168, 336}		{24, 48, 96, 288}
d_m	{32, 64, 128, 256, 512}		
Number of Heads	{1, 2, 4, 8}		
Number of Encoder Layers	{1, 2, 3}		
Number of Decoder Layers	{1, 2, 3}		
Window Size	{4, 6, 8, 12}		
Conv Kernel Size	{3, 5, 7}		
Learning Rate (log domain)	$[10^{-4}, 10^{-2}]$		
Dropout	{0.05, 0.1, 0.15, 0.2, 0.3}		

We use *Pytorch Lightning* as the training framework. We select MSE as the loss function and apply the Adam optimizer [11] during the training of the model. For hyperparameter tuning, we define the ranges of different hyperparameters and adopt *Optuna* [2] to optimize parameters automatically. After tuning, we use the optimal parameters for evaluation. For illustration, Table 2 presents the ranges of hyperparameters for our WAG-NAT model.

4.3 Results and Analysis

Table 3 lists the experimental results of all methods on four datasets over test set. We define a short-term forecasting task when $\tau \leq 48$, otherwise a long-term forecasting task. Our WAG-NAT model wins most benchmarks in short- and long-term forecasting.

Short-Term Forecasting. As for short-term forecasting, we conclude that (1) Transformer outperforms WAG-NAT on ETTh1 and BMSAQ datasets, and is less effective than WAG-NAT on other datasets. (2) WAG-NAT shows significant

Table 3. Performance comparison of different models on test set of the 4 datasets. The best results across various benchmarks are **bolded**.

Model		WAG-NET		Informer†		LogTrans		TCN		Transformer		LSTMa	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	24[a]	0.473	0.479	0.577	0.549	0.686	0.604	0.472	0.501	**0.449**	**0.479**	0.650	0.624
	48	0.467	**0.486**	0.685	0.625	0.766	0.757	0.525	0.539	**0.466**	0.497	0.702	0.675
	168	**0.653**	**0.601**	0.931	0.752	1.002	0.846	0.743	0.670	0.714	0.633	1.212	0.867
	336	**0.857**	**0.731**	1.128	0.873	1.362	0.952	0.981	0.792	0.932	0.805	1.424	0.994
ETTh2	24	**0.261**	**0.384**	0.720	0.665	0.828	0.750	0.696	0.678	0.424	0.501	1.143	0.813
	48	**0.650**	**0.648**	1.457	1.001	1.806	1.034	0.836	0.964	1.522	0.949	1.671	1.221
	168	1.722	1.112	3.489	1.515	4.070	1.681	2.456	1.247	**1.622**	**0.973**	4.117	1.674
	336	**2.019**	**1.180**	2.723	1.340	3.875	1.763	3.096	1.496	2.662	1.219	3.434	1.549
ETTm1	24	**0.274**	**0.346**	0.323	0.369	0.419	0.412	0.598	0.504	0.315	0.389	0.621	0.629
	48	**0.334**	**0.387**	0.494	0.503	0.507	0.583	0.590	0.548	0.445	0.462	1.392	0.939
	96	**0.413**	**0.448**	0.678	0.614	0.768	0.792	0.468	0.497	0.464	0.482	1.339	0.913
	288	**0.547**	**0.551**	1.056	0.786	1.462	1.320	0.721	0.663	0.681	0.631	1.740	1.124
BMSAQ[b]	24	2.858	3.710	3.526	4.187	3.590	4.779	3.026	**3.376**	**2.735**	3.686	3.690	4.693
	48	3.338	4.357	3.557	4.455	3.900	4.679	3.710	4.529	**3.237**	**4.187**	4.147	5.174
	168	**3.595**	**4.651**	3.854	5.013	4.075	4.874	3.729	4.743	3.739	4.976	4.348	4.865
	336	**3.816**	**4.957**	4.168	5.235	4.245	5.010	4.184	5.191	4.250	5.059	5.214	5.454

†To be fair, we reimplement Informer to support entity embeddings for categorical inputs, as mentioned in Sect. 3.1.

[a] This column represents the forecasting horizon τ.

[b] Due to the diversity of normalization methods, the scale of metrics on BMSAQ dataset varies from other datasets. To align the numerical format, the scale of metrics is 10^{-3}.

improvement compared to the RNN-based model LSTMa with an average MSE decrease of 45.69% ($\tau = 24$) and 47.52% ($\tau = 48$), which means the attention mechanism extracts global information better, and is less likely to lose memories temporally as found in the case of RNNs. (3) Compared to the CNN-based model, namely TCN, WAG-NAT also displays better results, indicating local and global information is integrated efficiently with the window attention mechanism, and hence the effect is better than only extracting local information. (4) WAG-NAT surpasses the generative Transformers, i.e., Informer and LogTrans, which reveals that the proposed generator can decide optimal sequences as decoder inputs. Besides, it justifies the choice of queries, keys, and values of the multi-head attention module in the generator.

Long-Term Forecasting. With regard to long-term forecasting, we suggest (1) the RNN-based LSTMa achieves the worst results across benchmarks, revealing that long-range dependency is lost when the memory path gets longer substantially. (2) Concerning the vanilla Transformer, the cumulative error enlarges as τ increases, and thus becomes a bottleneck of the autoregressive Transformer. (3) Compared with generative Transformers, WAG-NAT gains an MSE decrease of 31.91% ($\tau = 168$) and 25.56% ($\tau = 336$) on the ETTh1, ETTh2 and BMSAQ datasets on average, demonstrating the mechanism of the proposed generator is more capable and aligned.

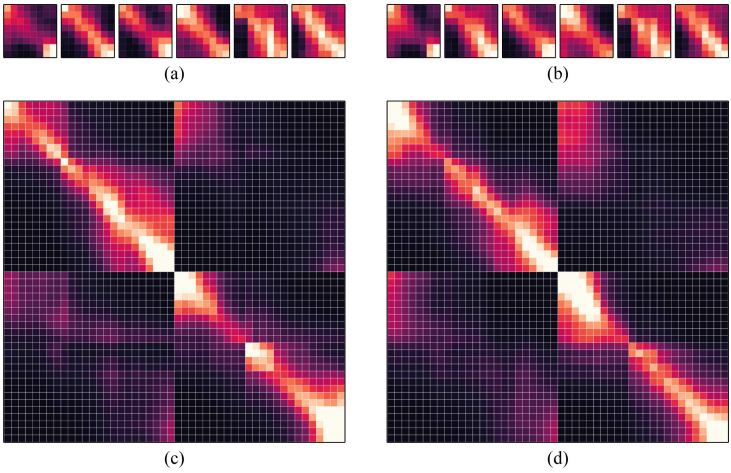


Fig. 3. Heatmaps of attention scores of WMSA and MSA after softmax activation at encoder layer 1. (a) and (b) represents attention scores of WMSA from window 1 to 6 at head 1 and 2 respectively. In the same way, (c) and (d) denotes attention scores of MSA at head 1 and 2. The brighter the square, the greater the corresponding attention score.

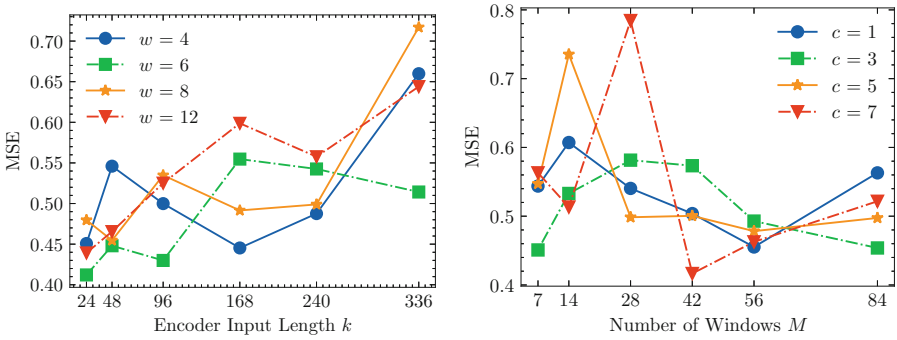
4.4 Effectiveness of Window Attention Mechanism

To illustrate the effectiveness of the window attention mechanism, we construct a WAG-NAT model with two encoder layers on the ETTh1 dataset where $k = 48$ and $w = 8$. Meanwhile, we replace the WMSA module of the encoder in WAG-NAT with MSA to investigate the distribution of different attention scores. Figure 3 presents the attention scores of {head 1, head 2} at encoder layer 1. From Fig. 3(c) and (d), we could observe that areas around the diagonal are brighter, which means most queries are prone to attend to neighboring keys. Besides, inactive queries suggest that redundant information is extracted, resulting in a waste of computing resources. Nevertheless, the attention scores of the WMSA module reveal that most queries in each window are active, which eliminates the capture of redundant information and further proves the rationality of the window attention mechanism.

4.5 Sensitivity Analysis

We perform the sensitivity analysis of the proposed WAG-NAT model on the ETTh1 dataset.

Window Size. Figure 4a presents the effect of window size $w \in \{4, 6, 8, 12\}$ on prediction accuracy at different encoder lengths $k \in \{24, 48, 96, 168, 240, 336\}$. The MSE tends to rise as k increases. Besides, a larger window size ($w = 8, 12$) performs worse prediction than a small window size, especially when input



(a) Window Size. (b) Kernel Size.

Fig. 4. Results of parameter sensitivity benchmarks.

sequence are long. Instead, $w = 6$ is a preferred value, whether for short-term or long-term forecasting tasks.

Kernel Size. Furthermore, to investigate the most efficient approach to interacting among windows under different window numbers M, we carry out several sets of experiments with different kernel sizes $c \in \{1, 3, 5, 7\}$. Specifically, we fix $k = 336$ and gradually increase $w \in \{4, 6, 8, 12, 24, 48\}$, then we get $M \in \{84, 56, 42, 28, 14, 7\}$. From Fig. 4b, we observe that MSE tends to decrease when M grows, which indicates that better results could be achieved when more windows are involved in the interaction. In addition, we find that the smaller the kernel size becomes, the greater accuracy of the prediction results. To conclude, local information is fully extracted, and compact window interaction yields better results. Surprisingly, even if $c = 1$, it works very well in most cases, which may indicate that adequate capture of the local patterns is a more important factor for time series forecasting.

5 Conclusion

In this work, we proposed a novel non-autoregressive Transformer model WAG-NAT, featuring a generator and window attention mechanism. The proposed window attention mechanism contains a window self-attention layer to achieve locality improvement and a window interaction layer to fuse local and global information from different windows. In addition, the one-step-forward prediction of WAG-NAT elevates the speed of inference. Results of various benchmarks on diverse real-world datasets demonstrate the capability and advancement of our model. Furthermore, extensive experiments were conducted to prove the effectiveness of the window attention mechanism and offer insights into parameter selection.

Acknowledgements. This work was supported in part by National Natural Science Foundation of China under Grant 61971467, in part by the Key Research and Development Program of Jiangsu Province of China under Grant BE2021013-1, in part by the Qinlan Project of Jiangsu Province.

References

1. Agarwal, O., Nenkova, A.: Temporal effects on pre-trained models for language processing tasks. Trans. Assoc. Comput. Linguist. **10**, 904–921 (2022)
2. Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: a next-generation hyperparameter optimization framework. In: KDD (2019)
3. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: ICLR (2015)
4. Bai, S., Kolter, J.Z., Koltun, V.: An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv:1803.01271 (2018)
5. Chen, K., Chen, G., Xu, D., Zhang, L., Huang, Y., Knoll, A.: NAST: non-autoregressive spatial-temporal transformer for time series forecasting. arXiv preprint arXiv:2102.05624 (2021)
6. Chen, Y., Chen, X., Xu, A., Sun, Q., Peng, X.: A hybrid CNN-transformer model for ozone concentration prediction. Air Qual. Atmos. Hlth. **15**(9), 1533–1546 (2022)
7. Gal, Y., Ghahramani, Z.: A theoretically grounded application of dropout in recurrent neural networks. In: NIPS (2016)
8. Gu, J., Bradbury, J., Xiong, C., Li, V.O.K., Socher, R.: Non-autoregressive neural machine translation. In: ICLR (2018)
9. Hewage, P., Trovati, M., Pereira, E., Behera, A.: Deep learning-based effective fine-grained weather forecasting model. Pattern Anal. Appl. **24**(1), 343–366 (2021)
10. Ioannou, Y., Robertson, D., Cipolla, R., Criminisi, A.: Deep roots: improving CNN efficiency with hierarchical filter groups. In: ICCV (2017)
11. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: ICLR (2015)
12. Koprinska, I., Wu, D., Wang, Z.: Convolutional neural networks for energy time series forecasting. In: IJCNN (2018)
13. Li, S., et al.: Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In: NIPS (2019)
14. Liu, Z., et al.: Swin transformer: hierarchical vision transformer using shifted windows. In: ICCV (2021)
15. Noh, S.H.: Analysis of gradient vanishing of RNNs and performance comparison. Information **12**(11), 442 (2021)
16. Nosratabadi, S., et al.: Data science in economics: comprehensive review of advanced machine learning and deep learning methods. Mathematics **8**(10), 1799 (2020)
17. Salman, A.G., Kanigoro, B., Heryadi, Y.: Weather forecasting using deep learning techniques. In: ICACSIS (2015)
18. Woschank, M., Rauch, E., Zsifkovits, H.: A review of further directions for artificial intelligence, machine learning, and deep learning in smart logistics. Sustainability **12**(9), 3760 (2020)
19. Wu, N., Green, B., Ben, X., O'Banion, S.: Deep transformer models for time series forecasting: the influenza prevalence case. arXiv preprint arXiv:2001.08317 (2020)
20. Zhou, H., et al.: Informer: beyond efficient transformer for long sequence time-series forecasting. In: AAAI (2021)