ESANN 2023 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 4-6 October 2023, i6doc.com publ., ISBN 978-2-87587-088-9. Available from http://www.i6doc.com/en/.

On Transformer Autoregressive Decoding for Multivariate Time Series Forecasting

Mohammed Aldosari and John A. Miller

School of Computing - University of Georgia
415 Boyd Research and Education Center, Athens, GA 30602

Abstract. The success of the Transformer model has promoted recent advances in time series forecasting. This adoption sparked an interest in developing efficient Transformer models that scale well for forecasting long sequences. This involves maintaining non-autoregressive one-time decoding. However, the role of autoregressive decoding is less explored. To address this gap, we revisit an essential idea of the vanilla Transformer model and show that autoregressive decoding works well compared to non-autoregressive decoding. It also becomes vital for critical forecasting tasks, such as pandemic forecasting, where the stakes are high. Our code and data are publicly available at `https://github.com/maldosari1/ar_transformer_tf`.

1 Introduction

Time series forecasting has been a long-standing problem in many application domains, including weather, energy consumption, and, more recently, pandemic forecasting. To tackle such a problem, one is equipped with a wide range of modeling techniques, including statistical models, neural models, or a hybrid of both. One excelling class of the neural models is the Transformer-based models, which work surprisingly well for modeling long-range sequences [1, 2]. The flexibility of the Transformer model made it an inevitable choice for forecasting. The mainstream Transformer-based models typically follow an encoder-decoder architecture where the encoder processes the past lagged values of a time series, and the decoder generates the forecasting of future horizons using non-autoregressive one-time generative decoding for time efficiency purposes [1]. It only uses temporal and positional embeddings of the target horizon length to generate the forecasting horizons at once using the observed data only. In our work, we seek to explore autoregressive decoding further and investigate its forecasting performance. This decoding uses the forecasted values while utilizing some actual values during training using a Teacher Forcing ratio. Such decoding with teacher forcing only uses the forecasted values at inference time with no access to actual values. Our empirical experiments show that autoregressive decoding works well compared to the more recent refinements to the vanilla Transformer utilizing segments-wise attention of the segmented input and output time series replacing the more computationally expensive point-wise attention. Our experiments also show an improvement in forecasting performance for three critical application domains: COVID-19, Influenza Like Illness (ILI), and Electricity Transformer Temperature (ETTh1) forecasting. Such an observation motivates a reconsideration of autoregressive decoding, which is vital for accurate forecasting.

2 Related Work

Recent extensive efforts have been dedicated to exploring the Transformer limitations and improving its forecasting abilities. For instance, the Informer [1] model proposes a sparse self-attention mechanism for addressing the quadratic time complexity of the vanilla Transformer model. The Autoformer [2] model decomposes the time series into trend and seasonality and applies series-wise attention instead of point-wise attention using autocorrelation. These works use non-autoregressive one-time decoding replacing autoregressive decoding of the vanilla Transformer. In addition, the role of the recursive versus direct forecasting has been well-studied from the statistical perspective [3]. In this sense, the recursive forecasting uses previous forecasts to generate the new forecasts by estimating the parameters of a single model and adjusting such estimates for each forecasting horizon [4]. The direct forecasting uses the observed data only to generate the future horizon by estimating a single model for each forecasting horizon independently. The recursive strategy tend to generate biased forecasts while the direct strategy tend to generate forecasts with high variance [3].

3 The Autoregressive Transformer

The time series forecasting problem can be formulated as a sequence-to-sequence learning problem. To model such a learning problem, one can easily adopt an encoder-decoder architecture, including the encoder-decoder Transformer model. The foundational building block of the Transformer model is the self-attention mechanism which aims to relate different parts of a given sequence to generate a refined representation of the entire sequence [5]. The self-attention mechanism projects a given sequence into three different representations: keys, queries, and values. The aim is to refine these randomly generated representations using backpropagation to better attend to the important information of the sequences. The attention mechanism is formulated in this equation $\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$, where Q represents the queries, K^T represents the keys, and V represents the values. The final representation of a value in the sequence is obtained as a weighted average of the Softmax scores multiplied by all the other values in the entire sequence. One can use the Transformer in different ways according to the desired task: Transformer-Encoder only to learn a contextualized language representation with no need for the decoder component or Transformer Encoder-Decoder for encoding and generation as in the time series problem.

Model Structure Given a time series forecasting problem with multiple covariates D, the model uses an embedding of all past covariates to forecast multivariate future horizons. Specifically, the model uses $x_{1:\tau} \in \mathbb{R}^{\tau \times D}$ to forecast $x_{\tau+1:T} \in \mathbb{R}^{T \times D}$ where τ represents the number of lagged values, T represents the number of future forecasts, and D is the number of covariates following the same problem definition as in [6].

Encoder The encoder follows the core encoder architecture of the vanilla Transformer model applied to the segmented input time series. The encoder focuses on modeling the segment-wise temporal behaviors of the past lagged values instead of the series-wise attention which can be computationally expensive for long sequences. The encoder segments the input time series into non-overlapping segments using a fixed segment length and applies the multi-headed self-attention between these segments. The input to the encoder is lagged values $x_{enc} = x_{1:\tau} \in \mathbb{R}^{\frac{\tau}{segment\_len} \times D \times d\_model}$. The encoder also uses positional and temporal embeddings being added to the covariates embeddings.

Decoder The decoder is similar to the vanilla Transformer decoder applied to the segments used for generation. To begin generation the target time series, the decoder uses fixed last segments from the input time series along with only positional and temporal encodings of the remaining target time series length. Specifically, the decoder can be used in three ways:

(1) a non-autoregressive one-time decoding which generates a target horizon without conditioning on actual values or the model's forecasts. The input to the non-autoregressive (NAR) decoder is a concatenation of a fixed number of segments taken from input sequence along with temporal and positional encodings of the target horizon. It is formulated as follows.

$$x_{dec} = Concat\{x^{-segments}, (Temp + PE) \in \mathbb{R}^{\frac{T}{segment\_len} \times D \times d\_model}\} \tag{1}$$

where $x^{-segments}$ is the start segments taken from the input time series, Temp is the temporal embeddings, and PE is the positional embeddings.

(2) an autoregressive decoder that generates a forecast at time t conditioned on the forecast from the previous time steps during training and inference. The input to the autoregressiv (AR) decoder is a concatenation of a fixed number of segments taken from the input sequence along with temporal and positional encodings of a single segment to generate the first target segment. The decoder then uses the newly forecasted segment when generating the next segments.

$$x_{dec} = Concat\{x^{-segments}, x_{dec\hat{+}embed} \in \mathbb{R}^{S_{t-1} \times D \times d\_model}\} \tag{2}$$

where $x^{-segments}$ is the start segments taken from the input time series, $x_{\hat{dec}}$ is the newly forecasted output segment, embed is the temporal and positional embeddings, and S_{t-1} is the segment at the previous time step.

(3) an autoregressive decoder that generates a forecast at time t conditioned on the previous forecast or on the actual value at the previous time steps during training only [7, 8]. The model uses its own forecasts during inference without access to actual values. The input to the autoregressiv (AR) with Teacher Forcing is a concatenation of a fixed number of segments from the input sequence along with temporal and positional encodings of a single segment to generate the first target segment. The decoder can choose to use the newly forecasted segment or the actual segment from the last time step for which a forecasting

ESANN 2023 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 4-6 October 2023, i6doc.com publ., ISBN 978-2-87587-088-9. Available from http://www.i6doc.com/en/.

has been obtained.

$$x_{dec} = Concat\{x^{-segments}, x_{\hat{dec+embed}} or x_{enc+embed} \in \mathbb{R}^{S_i \times D \times d\_model}\} \quad (3)$$

where $x^{-segments}$ is the start segments taken from the input time series, $x_{\hat{dec}}$ is the newly forecasted output segment along with the temporal and positional embeddings, x_{enc} is the actual segments from the last time step, and embed is the temporal and positional embeddings. The teacher forcing strategy allows the model to either use the actual values or use the model outputs from the previous time step. It improves the learning process by allowing the model to "stay close" to the observed target data [9] without propagating the compound errors far through the network. The state-of-the-art Transformer-based models utilize the first decoder design choice. In this work, we reconsider the last two design choices and show that autoregressive decoding well compared to non-autoregressive decoding. The encoder and decoder do not not model the cross-covariate interactions; however, they only models segment-wise interactions within each covariate. In order to be fed to the encoder and decoder, the time series segments are obtained using the Dimension-Segment-Wise (DSW) Embedding proposed by the Crossformer model.

4 Experiments

Datasets In this work, we consider three multivariate time series datasets: COVID-19, ILI, and ETTh1. The COVID-19 dataset is maintained by Our World in Data (OWID) [1] and contains daily reportings of cases related to SARS-CoV-2 of six covariates from December 14th, 2020, to June 18th, 2022. The ILI dataset is maintained by the Centers for Disease Control and Prevention (CDC)[2] and contains weekly reportings of patients with symptoms related to influenza in the United States of seven covariates from January 1st, 2002, to June 30th, 2020. The ETTh1 dataset is maintained by [1][3] and contains hourly reportings of electricity transformers of seven covariates from July 07th, 2016, to June 26th, 2018.

Evaluation To evaluate the forecasting performance, we use two main metrics: the mean squared error (MSE) and the mean absolute error (MAE). The first metric measures the squared error between the actual and forecasted values, and it is calculated using this equation MSE $= \frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)^2$ where y_t is the actual value, and $\hat{y}_t$ is the forecasted value. The second metric measures the mean absolute error between the actual and forecasted values, and it is calculated using this equation MAE $= \frac{1}{n}\sum_{t=1}^{n}|y_t - \hat{y}_t|$ where y_t is the actual value, and $\hat{y}_t$ is the forecasted value. The MSE and the MAE are unbounded metrics applied to the transformed data and model outputs with no scaling back to the original

[1] `https://github.com/owid/covid-19-data`
[2] `https://gis.cdc.gov/grasp/fluview/fluportaldashboard.html`
[3] `https://github.com/zhouhaoyi/ETDataset`

Table 1: MSE and MAE results for the Transformer based models. We show that the Autoregressive Transformer with Teacher-Forcing (AR-Transformer-TF) works well compared to the non-autoregressive transformers.

		NAR-Transformer		Informer		Autoformer		Crossformer		AR-Transformer		AR-Transformer-TF		
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	TF-Ratio
Covid Days	24	0.413	0.545	0.618	0.656	0.710	0.634	0.303	0.462	0.119	0.266	**0.086**	**0.220**	0.8
	36	0.717	0.733	0.976	0.819	0.391	0.476	0.465	0.586	0.338	0.388	**0.157**	**0.302**	0.8
	48	1.227	0.984	1.481	1.039	0.647	0.630	0.594	0.673	0.173	0.305	**0.139**	**0.280**	0.6
	55	1.441	1.070	1.682	1.115	0.662	0.641	0.571	0.653	0.243	0.354	**0.228**	**0.340**	0.2
ILI Weeks	24	4.475	1.376	4.856	1.453	4.037	1.443	4.095	1.332	**2.313**	**0.977**	2.256	0.998	0.4
	36	4.506	1.380	4.659	1.430	3.266	1.236	4.268	1.345	2.551	0.971	**2.249**	**0.872**	0.6
	48	4.682	1.404	4.659	1.449	3.242	1.226	4.158	1.351	2.524	0.960	**2.383**	**0.924**	0.6
	60	4.667	1.422	4.781	1.481	3.637	1.338	4.229	1.362	2.612	**0.982**	**2.568**	0.992	0.2
ETTh1 Hours	24	0.717	0.664	0.665	0.598	0.367	0.414	**0.307**	**0.361**	0.314	0.361	0.330	0.368	0.8
	48	0.987	0.799	0.783	0.671	0.471	0.465	0.358	0.400	**0.352**	0.385	0.355	**0.384**	0.6
	168	1.071	0.851	1.041	0.773	0.495	0.480	0.574	0.547	0.432	0.438	**0.424**	**0.434**	0.2
	336	1.154	0.884	1.130	0.830	0.508	0.486	0.655	0.583	0.466	0.465	**0.451**	**0.448**	0.6
	720	1.082	0.849	1.242	0.895	0.551	0.528	1.068	0.809	0.550	0.524	**0.528**	**0.515**	1.0

NAR: Non-autoregressive, AR: Autoregressive, and TF: Teacher Frorcing

scale following the previous work in the literature.

Implementation details We implement our neural models in Python by adopting and extending the implementation pipeline of the Autoformer and Crossformer models [2, 6]. We fix some parameters like reported by the baseline models. We set the past lagged values $\tau = 36$ and future horizon = $\{24, 36, 48, 55\}$ for the COVID and horizon = $\{24, 36, 48, 60\}$ for the ILI dataset. We also set $\tau = 96$ and horizon = $\{24, 48, 168, 336, 720\}$ for the ETTh1 dataset. We set encoder layers to 2 and decoder layers to 1, the model dimensions to 512, and the Transformer feedforward layer to 2048. We also set the batch size to 32 trained for 30 epochs with a early stopping patience set to 3. The number of dimensions D is 6 for the COVID data, 7 for the ILI and ETTh1 datasets. For the AR-Transformer-TF, we tuned the model using Wandb [4] to find the best teacher forcing ration over 5 different runs. All models are trained and evaluated using a fixed random seed.

Main Results To highlight the advantage of autoregressive decoding, we compare the autoregressive Transformer with other state-of-the-art Transformer-based models. We observe a consistent improvement of the autoregressive Transformer across the COVID-19, ILI, and ETTh1 datasets compared to the NAR-Transformer, Informer, Autoformer, and Crossformer models. We illustrate our results in table 1. For example, the autoregressive Transformer with teacher forcing archives a lower MAE and MSE across all forecasting horizons for the COVID-19, ILI, ETTh1 datasets. We show how adding teacher forcing can further improve the forecasting abilities of the autoregressive transformer. For instance, the MSE and MAE improved from (0.119, 0.266) to (0.086, 0.220) for forecasting 24 days in the COVID-19 dataset when utilizing teacher forcing strategy. We also observe that the teacher forcing ratio is a critical design choice that varies across sequence lengths and datasets as shown in table 1.

[4]Experiment tracking with weights and biases. `https://docs.wandb.ai/guides/sweeps`

5 Conclusion

In this work, we revisit the autoregressive decoding of the vanilla Transformer model and explore its forecasting performance. We show that the autoregressive decoding surpasses the state-of-the-art Transformer-based models resulting in more accurate forecasting. Our empirical experiments also show the effectiveness of using the teacher forcing in the vanilla autoregressive decoding on critical forecasting tasks: COVID-19, ILI, and ETTh1. The reported results will inspire further research on understanding the role of the Transformer autoregressive decoding in time series forecasting.

References

[1] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11106–11115, 2021.

[2] 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.

[3] Souhaib Ben Taieb, Rob J Hyndman, et al. *Recursive and direct multi-step forecasting: the best of both worlds*, volume 19. Department of Econometrics and Business Statistics, Monash Univ., 2012.

[4] Guillaume Chevillon. Direct multi-step estimation and forecasting. *Journal of Economic Surveys*, 21(4):746–785, 2007.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[6] Yunhao Zhang and Junchi Yan. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *The Eleventh International Conference on Learning Representations*, 2022.

[7] Chunqi Wang, Ji Zhang, and Haiqing Chen. Semi-autoregressive neural machine translation. *arXiv preprint arXiv:1808.08583*, 2018.

[8] Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. Non-autoregressive neural machine translation. *arXiv preprint arXiv:1711.02281*, 2017.

[9] Alex M Lamb, Anirudh Goyal ALIAS PARTH GOYAL, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. Professor forcing: A new algorithm for training recurrent networks. *Advances in neural information processing systems*, 29, 2016.