Toward Understanding Long Sequence Time Series Forecasting via Transformers

Ethan Wu, Donghan Yu, Ruohong Zhang, Yiming Yang

May 4, 2022

Abstract

Making long sequence forecasting on time series involves a number of modeling challenges. Nonetheless, Transformer architecture inspires recent progress in this domain. In this report, we focus on two major advancements: non-autoregressive decoding and Autoformer architecture. A number of experiments on real-world and synthetic datasets are performed to verify their superior performance and investigate their working mechanism. In addition, the benefits and drawbacks of different methods are discussed. We summarize our findings and interpretations in the final discussion. [1]

1 Introduction

Time series has long been a heated research domain for the machine learning community. Successful modeling of temporal patterns has profound applications in finance, logistics, natural sciences, etc. The forecasting task can be formulated as predicting observations in the future $\mathbf{X}_{\text{output}} \in \mathbb{R}^{L_{\text{target}} \times d}$ based on the observations up until now $\mathbf{X}_{\text{input}} \in \mathbb{R}^{L_{\text{input}} \times d}$. However, non-stationarity in many real-world scenarios complicates the task. Another challenge arises due to multivariate signal ($d > 1$), which entails the consideration of spatial relationship. Finally, long sequence forecasting (large L_{target}), which involves predictions for tens or hundreds of future timestamps, requires the ability to capture long-term dependencies.

The neural network approach to time series forecasting was once dominated by Recurrent Nueral Networks (Lai et al. [2017]). More recently, however, Transformer due to Vaswani et al. [2017] has created a new paradigm for the deep learning community. The architecture has achieved record-breaking successes in a number of Computer Vision and Natural Language Processing tasks. A subsequent question is how we can adopt this architecture to tackle challenges in time series forecasting. Li et al. [2019] introduces the LogSparse Transformer to reduce the asymptotic cost of the multihead attention. The Informer due to Zhou et al. [2020] leverages non-autoregressive decoding and renovates the attention mechanism again. By injecting stronger inductive biases, Wu et al. [2021] replaces the attention and

[1] The code is available at: https://github.com/yongyi-wu/lstf.

layer normalization (Ba et al. [2016]) modules by autocorrelation and decomposition modules, respectively. While retaining the decomposition scheme, FEDformer by Zhou et al. [2022] shifts to modeling frequency-domain signals instead.

Of all recent progress, Informer (Zhou et al. [2020]) and Autoformer (Wu et al. [2021]) have shown most dramatic improvement compared to previous methods, which motivate the experiments presented in this report. In particular, we conduct comprehensive ablation studies on real and synthetic datasets to:

- Verify the superior performance of Informer's non-autoregressive decoding versus the autoregressive counterpart;
- Investigate the inductive biases in Autoformer's autocorrelation module and decomposition module.

2 Non-Autoregressive Decoding

2.1 Architecture

Informer (Zhou et al. [2020]) is the first Transformer variant that exploites non-autoregressive decoding (termed *generative inference* in the paper) for time series forecasting. The key idea is to predict L_{target} future timestamps in just one forward pass and thus save inference time. In this section, we focus on vanilla Transformers (Vaswani et al. [2017]) with different decoding schemes.

2.1.1 Multihead Attention

The attention mechanism follows the vanilla multihead attention. Within each head, the output is computed by

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}})\mathbf{V}.$$

Overall, the output is $\text{Multihead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = [\text{head}_1 \| \text{head}_2 \| \cdots \| \text{head}_h]\mathbf{W}^O$, where we define $\text{head}_i = \text{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V)$. Here, $\|$ is the concatenation operator and $\mathbf{W}$'s are projection matrices of appropriate shapes.

2.1.2 Decoder

The autoregressive decoder follows the design of seq2seq Transformer models. In a long sequence forecasting setting, when predicting values $X_{j,*} \in \mathbb{R}^d$ at timestamp j, we concatenate all previous predictions $\hat{\mathbf{X}}_{<j,*} \in \mathbb{R}^{(j-1)\times d}$ and feed them into the model. Notice that this decoding scheme is inherently sequential and hence can be extremely slow during inference. Moreover, the model is vulnerable to the training-testing distribution shift due to teacher-forced training (Williams and Zipser [1989]).

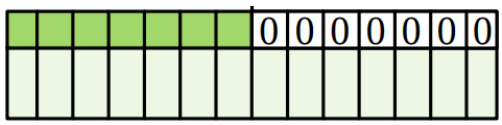

Figure 1: Input to a non-autoregressive decoder.

On the other hand, non-autoregressive decoder forecasts in parallel. The layout of decoder input is shown in Figure 1, where the dark green blocks are (optionally) previous observations that create *context* for decoding, the white zero blocks are placeholders where predictions will be made, and the light green blocks are positional and temporal embeddings which are intact. Finally, a triangular causal mask is applied to decoder's self attention module so that each timestamp can attend to previous observations only.

2.2 Experiments

While the Informer paper (Zhou et al. [2020]) does contain an ablation study on the non-autoregressive decoding, only two prediction lengths L_{target} are examined and the authors did not decouple the influence of the noval attention module. To verify the superiority of a non-autoregressive decoder, we run more experiments on six real-world benchmarks and observe expected results.

Interestingly, we also notice that the decoder taking input as Figure 1 is sufficient to perform forecasting. Therefore, we remove encoder and evaluate the `Decoder-Only` model as well.

2.2.1 Datasets

We use six real-world dataset that are popular in time series research community:

- `ETT`(Zhou et al. [2020]): This dataset is related to electric power deployment. The data is 7-dimensional plus a timestamp. The sampling rate of `ETTh1` is 1 hour and of `ETTm2` is 15 minutes. `ETTh1` has 14k observations in total, whereas `ETTm2` has 70k observations in total.

- `Electricity`[2]: It contains electricity consumption of 321 clients. The sampling rate is 1 hour. There are 26k observations in total.

- `Exchange`(Lai et al. [2017]): It documents daily exchange rate across 7 countries. There are 8k observations in total.

- `Traffic`[3]: This is a collection of road occupancy rates of 862 San Francisco Bay area freeways. The sampling rate is 1 hour. There are 18k observations in total.

[2]https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014
[3]http://pems.dot.ca.gov/

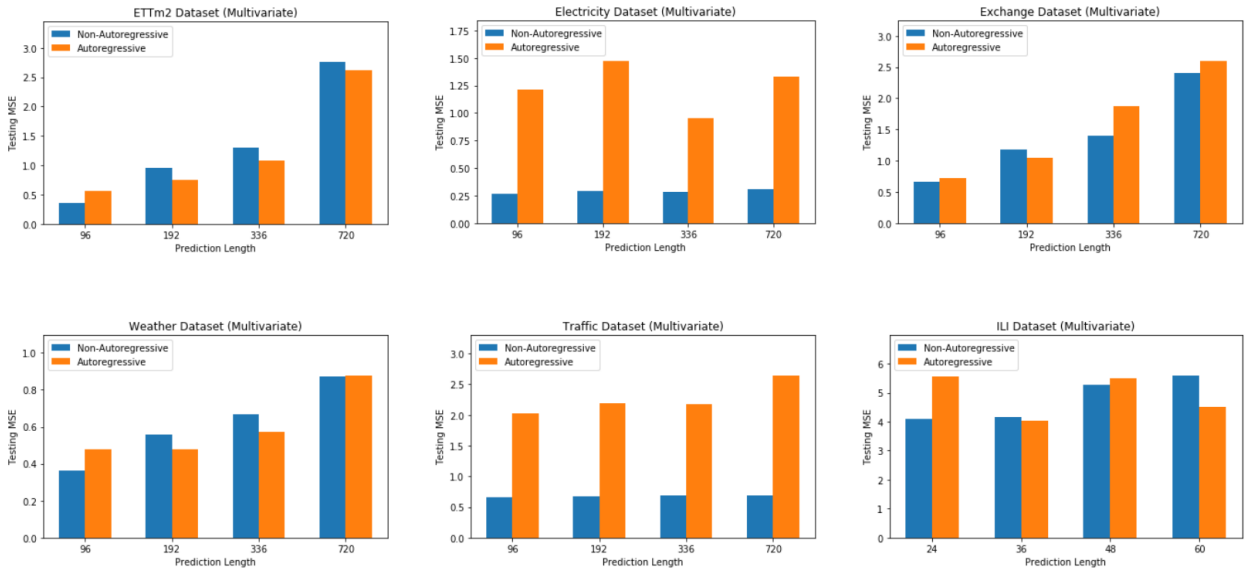

Figure 2: Mean square errors of autoregressive versus non-autoregressive decoding.

- `Weather`[4]: This is local climatological data from 2010 to 2013 of the United States. The data is 20-dimensional plus a timestamp. The sampling rate is 1 hour. There are 50k observations in total.
- `ILI`[5]: The dataset contains the percentage of patients with influenza-like illness from 2002 to 2021. The data is 7-dimensional plus the timestamp with a 7-day sampling rate. There are 967 observations in total.

2.2.2 Results

Figure 2 shows the mean square errors of forecasting results on different test datasets with varied prediction lengths. The lower the bar, the better the model. Overall, non-autoregressive decoding performs on par with, if not better than, autoregressive decoding. The gap is especially large in `Electricity` and `Traffic` datasets. A possible reason is that time series forecasting relies more on an understanding of large-scale pattern; instead, the dependencies on previous one timestamp or two are rather weak. Therefore, autoregression does not bring about significant advantage.

Taking the inference speed into account, Figure 3 indicates that autoregressive decoding is an undisirable choice for sure. Interestingly, while Zhou et al. [2020] claims that Informer improves the asymptotic cost of the attention mechanism, empirically vanilla attention with non-autoregressive decoding takes even smaller amount of time. This result highlights the discrepancy between the asymptotic analysis and wall-clock time. Meanwhile, it reinforces

[4]https://www.bgc-jena.mpg.de/wetter/
[5]https://gis.cdc.gov/grasp/fluview/fluportaldashboard.html

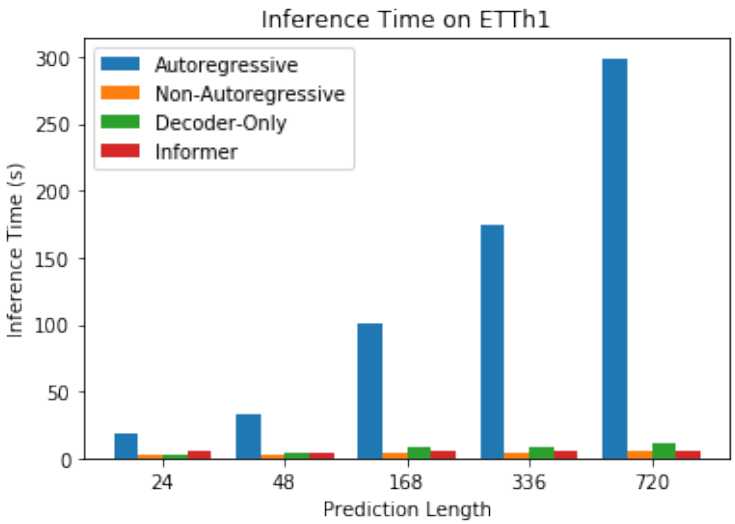

Figure 3: Inference speed measured by second on the `ETTh1` test dataset.

the necessity of incorporating non-autoregressive decoding scheme into Transformer variants for the long sequence time series forecasting.

As mentioned previously, we also evaluate a vanilla Transformer without an encoder, since the non-autoregressive decoder is theoretically with contextualized input (Figure 1) is sufficient in this setting. Figure 4 compares results with state-of-the-art Transformer variants. Although the `Decoder-Only` model has achieved competitive performance against Informer, it falls short of more recent models equipped with the decomposition module. Moreover, we found empirically that `Decoder-Only` is insensitive as we scale up the model parameters. Thus, while removing the encoder can be a promising way to reduce model size robustly, it fails to generate significant improvement. The hope lies in the architecture innovation.

3 Autoformer Architecture

3.1 Architecture

Autoformer (Wu et al. [2021]) makes substantial modifications to the building blocks of Transformer. As shown in Figure 5, the multihead attention module is replaced by the autocorrelation module, and layer normalization module by decomposition module. The idea is to emulate the traditional seasonal-trend decomposition (Cleveland et al. [1990]) while leverage the flexibility of neural networks to model periodic patterns.

3.1.1 Decomposition Module

The module aims at decomposing the input signal $X \in \mathbb{R}^L$ into trend $T \in \mathbb{R}^L$ and seasonality $S \in \mathbb{R}^L$ such that $X = T + S$. The trend is expected to contain long-term non-stationary

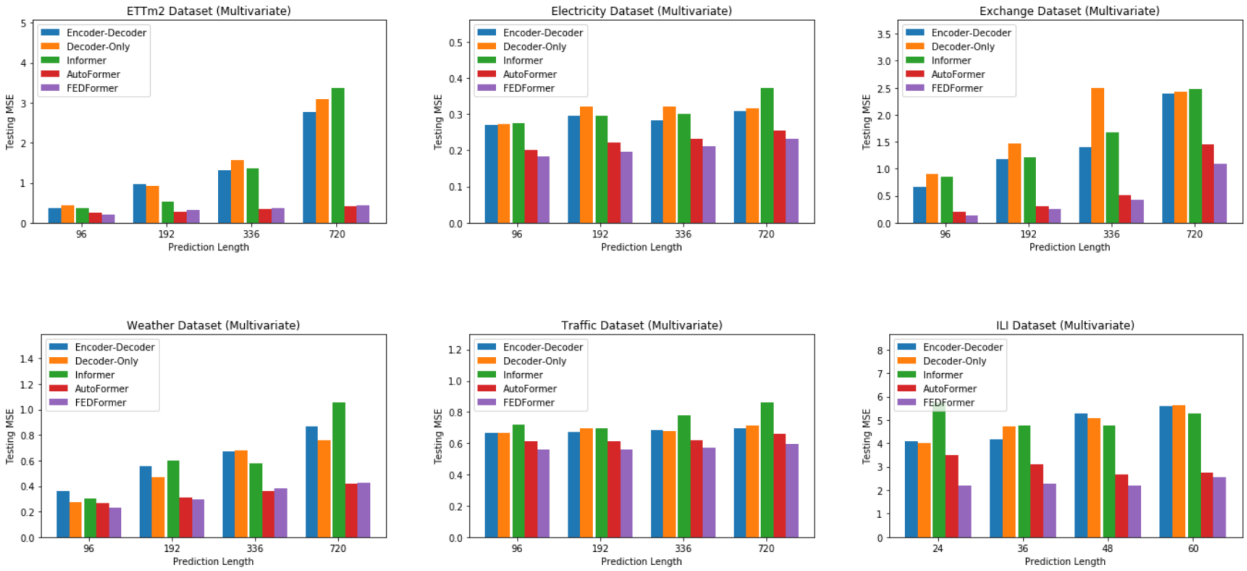

Figure 4: Mean square errors of Transformers with and without encoder as well as state-of-the-art variants.

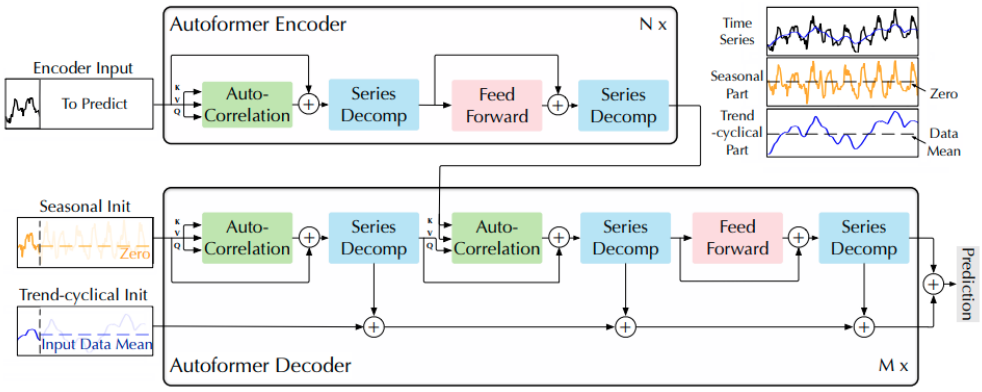

Figure 5: A visualization of Autoformer architecture.

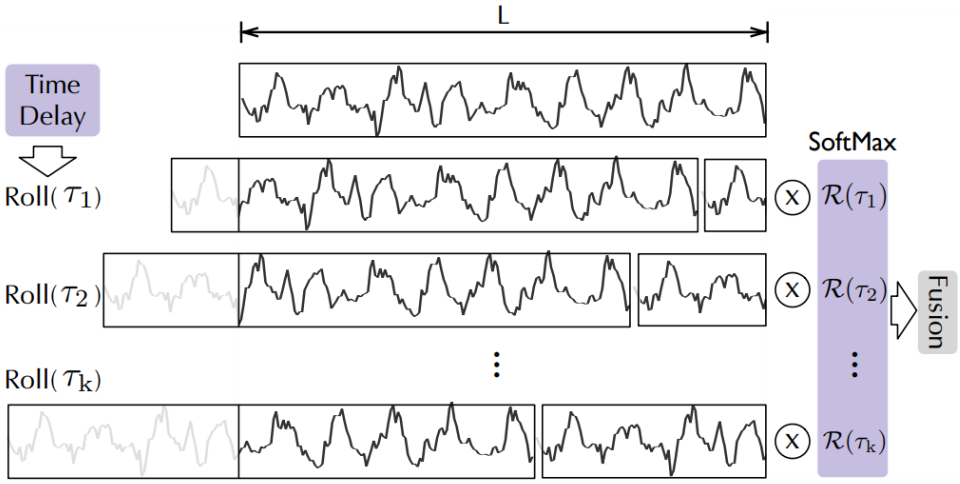

Figure 6: A visualization of autocorrelation module.

changes, whereas seasonality is more concerned about cyclical patterns. To this end, we just compute moving average (i.e. perform average pooling) along the temporal domain to obtain trend, and substact it element-wise from the input signal to get seasonality. Mathematically, we have

$$T = \text{AvgPool}(X) \qquad\qquad S = X - T.$$

Notice in Figure 5 that the input into the decoder gets decomposed in the beginning, which, we hypothesize, gives an appropriate *baseline* for later forecasting. As noted by Wen et al. [2022], the decomposition mechanism indeed boosts all Transformer variants' performance significantly. These ideas will be revisited in our experiments.

3.1.2 Autocorrelation Module

This module emulates the idea of autocorrelation to compute sequence-level similarities in hope of capturing period-based dependencies. More formally, the similarity between a query sequence $Q \in \mathbb{R}^L$ and a τ-delayed key sequence $K \in \mathbb{R}^L$ is defined as

$$R_{Q,K}(\tau) = \sum_{t=1}^{L} Q_t K_{t-\tau}.$$

The intuition is that $R_{Q,K}(\tau)$ will be large in magnitude if two sequences are similar and have period τ. To optimize memory and speed consumptions, only top-k different τ's that give high $R_{Q,K}(\tau)$ are selected and normalized by `softmax` to be attention weights $\{\alpha_{\tau_i}\}_{i=1}^k$. Finally, the output is the linear combination of k different wrap-arounded (i.e. `Roll`-ed) value sequences $V \in \mathbb{R}^L$: $\sum_{i=1}^k \alpha_{\tau_i} \text{Roll}(V, \tau_i)$ (see Figure 6).

For efficient computation of $R_{Q,K}(\tau)$, the model employs the Fast Fourier Transforms. We refer the interested readers to the original paper for more details.

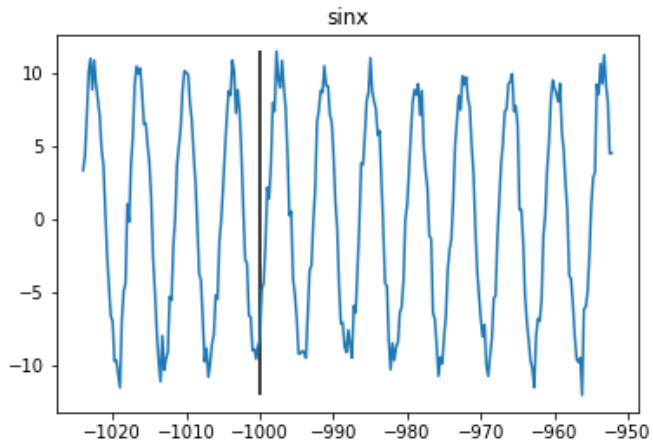

Figure 7: A subsequence of the entire `sinx` dataset. To the left of the black vertical line is fed into the encoder, while to the right is to be predicted. The encoder *should* be able to discover the periodicity.

3.2 Experiments

The experiments are designed to explain the superior performance brought about by the decomposition module and autocorrelation module, respectively, and investigate their robustness under different regularities. On top of real-world benchmarks, we generate nine synthetic datasets of various characteristics. Except stated explicitly, the models are trained and evaluated under the same hyperparameter setting.

3.2.1 Datasets

To avoid complicating the analysis, all synthetic datasets are univariate with the same length, that is, 8k timestamps. We take one periodic dataset, `sinx`, as an example to illustrate the data generation process. To begin, we initialize a equally spaced sequence

$$X = [\cdots, -0.75, -0.50, -0.25, 0., 0.25, 0.50, 0.75, \cdots].$$

Then, we feed it into some pre-defined function (`sin` in this case). Finally, Gaussian noise with zero mean and appropriate standard deviation is added. Importantly, we make the underlying period (if any) so small that the length of the encoder input (see Figure 7) is sufficient to discover periodicity.

Our synthetic datasets can be categorized into trended (i.e. containing underlying trend) and untrended (i.e. oscilating around 0). As for the oscilation, one underlying cycle corresponds to approximately 25 timestamps or length of 6 on the real line. Denoting standard Gaussian noise by $\mathcal{N}$, we present an overview of different datasets as follows. Readers may proceed to Appendix A for visualizations of all datasets.

- Trended

- `x`: $Y = X + \mathcal{N}$. This is just a canonical trend.
- `sinx_x`: $Y = 10 \sin X + X + \mathcal{N}$. This additively combines the canonical trend with a cyclical pattern. Signal-trend decomposition is supposed to work well in this case.
- `sinx_sqrtx`: $Y = 10 \sin X + 20\sqrt{X - \min(X)} + \mathcal{N}$. The trend is non-linear and, more specifically, concave downward.
- `sinx_x2_sym`: $Y = 10 \sin X + (\frac{X}{50})^2 + \mathcal{N}$. The trend is symmetric and concave upward.
- `sinx_x2_asym`: $Y = 10 \sin X + (\frac{X - \min(X)}{30})^2 + \mathcal{N}$. The trend is assymetric and concave upward. The training dataset and test dataset have drastically different means.

- Untrended
  - `sinx`: $Y = 10 \sin X + \mathcal{N}$. This is just a canonical periodic pattern. Autocorrelation module should be able to capture it.
  - `xsinx`: $Y = e^{X \mod 4} \dot{(} 10 \sin X + \mathcal{N})$. The amplitude varies over time.
  - `sinx_sin2x_sin4x`: $Y = 10 \sin X + 10 \sin 2X + 10 \sin 4X + \mathcal{N}$. The periodic pattern is more complicated.
  - `sinx_c`: $Y = 10 \sin X + (-1)^{\mathbb{I}[X \mod 16 < 8]} 30 + \mathcal{N}$. Each subsequence is shifted either upward or downward, requiring greater flexibility to capture these drastic changes.

Meanwhile, we also perform ablation studies on the aforementioned benchmarks. Detailed dataset descriptions can be found in Section 2.2.1.

3.2.2 Results

According to Figure 8, when there is an underlying trend, incorporating decomposition module improves the model performance significantly. Instead, if we naively employ autocorrelation module without a proper detrending process, the model can get even more confused. On the other hand, if there is no trend in the first place, decomposition module can in fact create negative impacts, possibly by introducing random noise.

Figure 9 plots the forecasting results of a randomly chosen subsequence. Plots for other datasets can be found in Appendix B. As expected, the decomposition module *recalibrate* the prediction baseline to an appropriate level, so the final results would not deviate too much from the target.

Since decomposition module simply computes the running average, the size of sliding window is a hyperparameter. Recall that by construction, one period corresponds to approximately 25 timestamps. Thus, we may expect models with sliding window size an integer multiple of 25 to achieve better performance. Figure 10 indeed confirms our expectation and emphasizes the importance of choosing an appropriate window size.

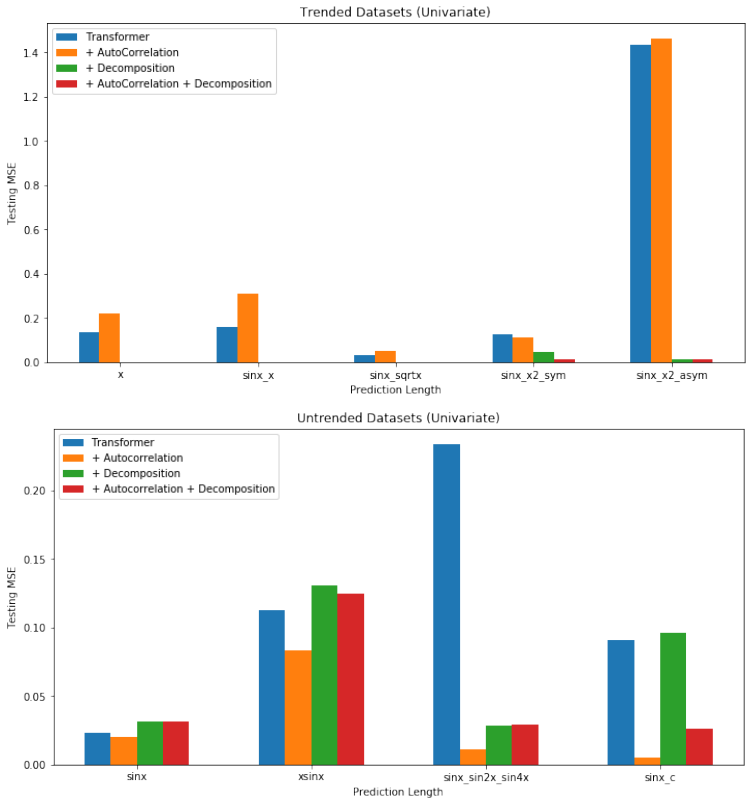

Figure 8: Mean square errors on trended and untrended synthetic datasets. The `+ AutoCorrelation + Decomposition` case is equivalent to the Autoformer.

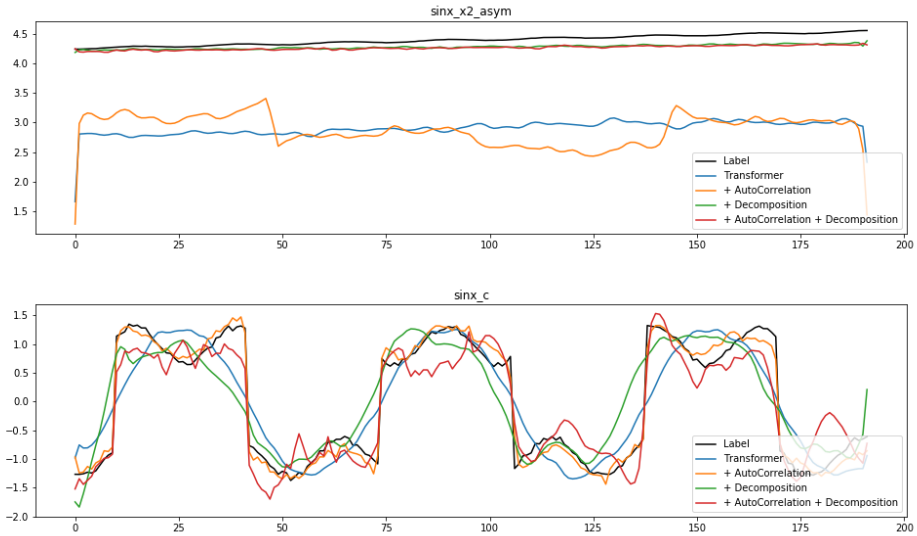

Figure 9: Forecasting plots of a random subsequence in one trended and one untrended synthetic datasets.

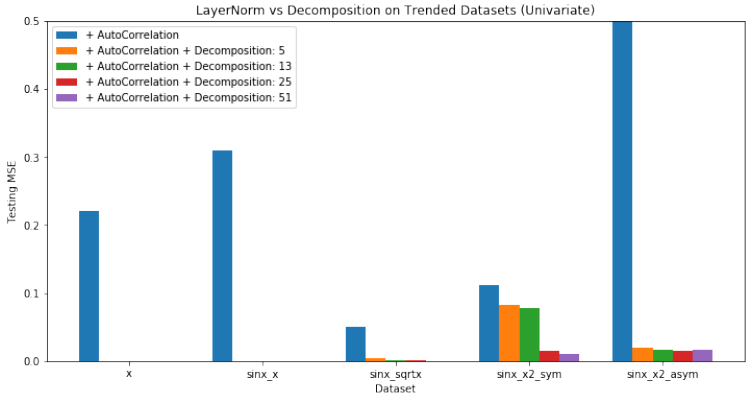

Figure 10: Mean square errors of models with varying decomposition window sizes.

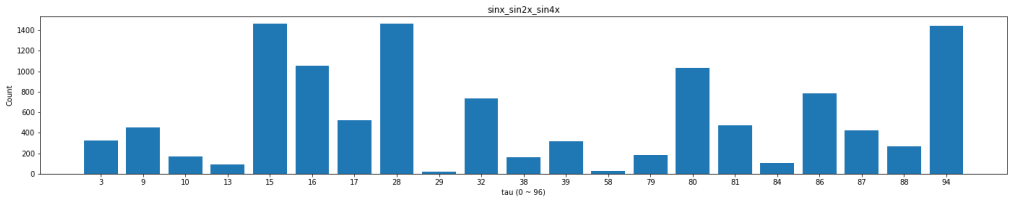

Figure 11: For each τ, the number of times $R_{Q,K}(\tau)$ is a top-k largest one.

As for the autocorrelation module, we notice that as long as the signal has been detrended, it is capable of improving the forecasting quality. This phenomenon is especially pronounced when we know beforehand that the data is untrended, so no decopmosition is performed to avoid introducing noise.

Furthermore, we can explore the explainability of autocorrelation weights, just like the attention weights. We expect that the top-k most significant τ's will include an integer multiple of 25 very often. Unfortunately, Figure 11 shows a case where the distribution of τ does not align with our intuition. More examples can be found in Appendix C. Most of the time, the distribution of τ does not seem to offer insight into the underlying periodicity of the data. Therefore, the working mechanism of the autocorrelation module appears to be a black box awaiting for future discovery.

Finally, we include the results on real-world datasets. The baseline model is vanilla Transformer and we replaces the attention module by autocorrelation module and/or layer normlization module by decomposition module. Consistent with findings due to Wu et al. [2021], the greatest empirical improvement comes from the decomposition module. However, the effect of autocorrelation module is more nuanced. This is possibly because the trend is harder to be identified and removed, and the underlying cyclical patterns are more complicated. While harder to interpret, the prediciton plots is also available in Appendix B.

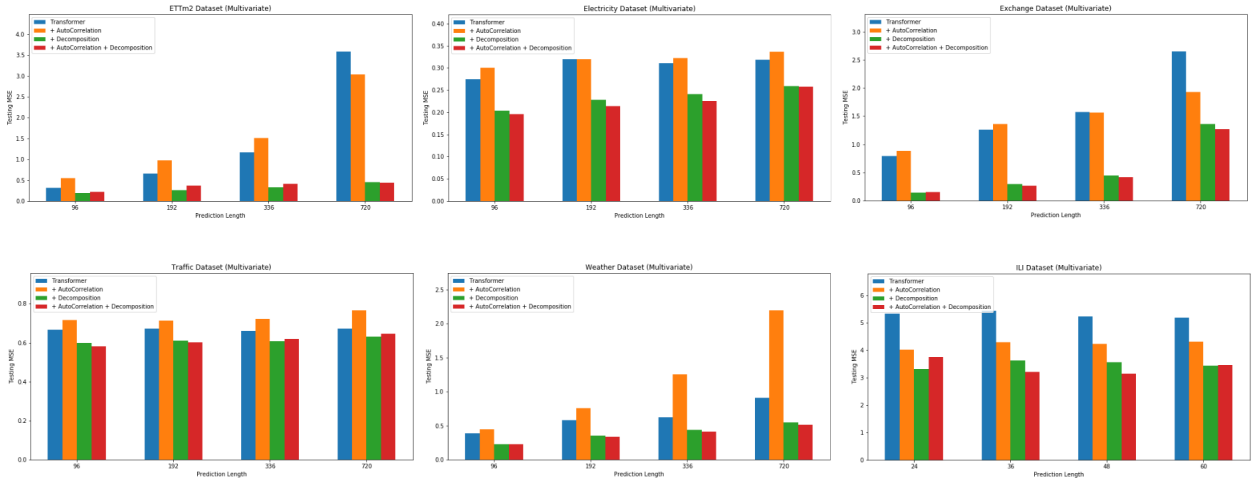

Figure 12: Mean square errors vanilla Transformer and its variants on real-world datasets.

4 Discussion

This report reviews the recent progress of using Transformer and its variants on the time series forecasting task. In particular, we focus on non-autoregressive decoding, seasonal-trend decomposition module and autocorrelation module, which empirically yields largest performance gains, and conduct comprehensive probing experiemnts to understand their working mechanism.

Shifting from autoregressive decoding to non-autoregressive decoding retains the original performance. That the shift is not as easy in conventional Natural Langauge Processing tasks as in this case may due to different statistical regularities in natural languages and time series. Nonetheless, the empirical results suggest non-autoregressive method as a baseline for architectural innovations. In contrast, the asymptotic improvement of attention mechanism creates little, if any, speedup, possibly because of the introduced overhead.

Despite non-parametric and seemingly naive, the decomposition module dramatically improves various Transformers across synthetic and real-world datasets. Still, it is not perfect. According to results on the synthetic datasets, this module may introduce noise into inherently clean and non-stationary data; also, it takes a hyperparameter which is hard to determine analytically in real world scenarios. With that being said, the ability to recalibrate the level at which predictions are performed makes it a strong baseline.

Autocorrelation module performs sequence-level attention, replacing the vanilla token-level attention. Although it achieves respectable performance gain on untrended synthetic datasets, it is vulnerable to the interference of trended signals. Furthermore, the autocorrelation weights lack the explainability as its attention counterpart.

In conclusion, non-autoregressive decoding is worth to be considered as a baseline approach. Besides, Autoformer's successes suggest future work improving the detrending mechanism and enhancing the sequence-level attention's robustness.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016. URL https://arxiv.org/abs/1607.06450.

Robert B. Cleveland, William S. Cleveland, Jean E. McRae, and Irma Terpenning. Stl: A seasonal-trend decomposition procedure based on loess (with discussion). *Journal of Official Statistics*, 6:3–73, 1990.

Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long- and short-term temporal patterns with deep neural networks, 2017. URL https://arxiv.org/abs/1703.07015.

Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting, 2019. URL https://arxiv.org/abs/1907.00235.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017. URL https://arxiv.org/abs/1706.03762.

Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. Transformers in time series: A survey, 2022. URL https://arxiv.org/abs/2202.07125.

Ronald J. Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280, 1989. doi: 10.1162/neco.1989.1.2.270.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting, 2021. URL https://arxiv.org/abs/2106.13008.

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, 2020. URL https://arxiv.org/abs/2012.07436.

Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting, 2022. URL https://arxiv.org/abs/2201.12740.

A Synthetic Datasets

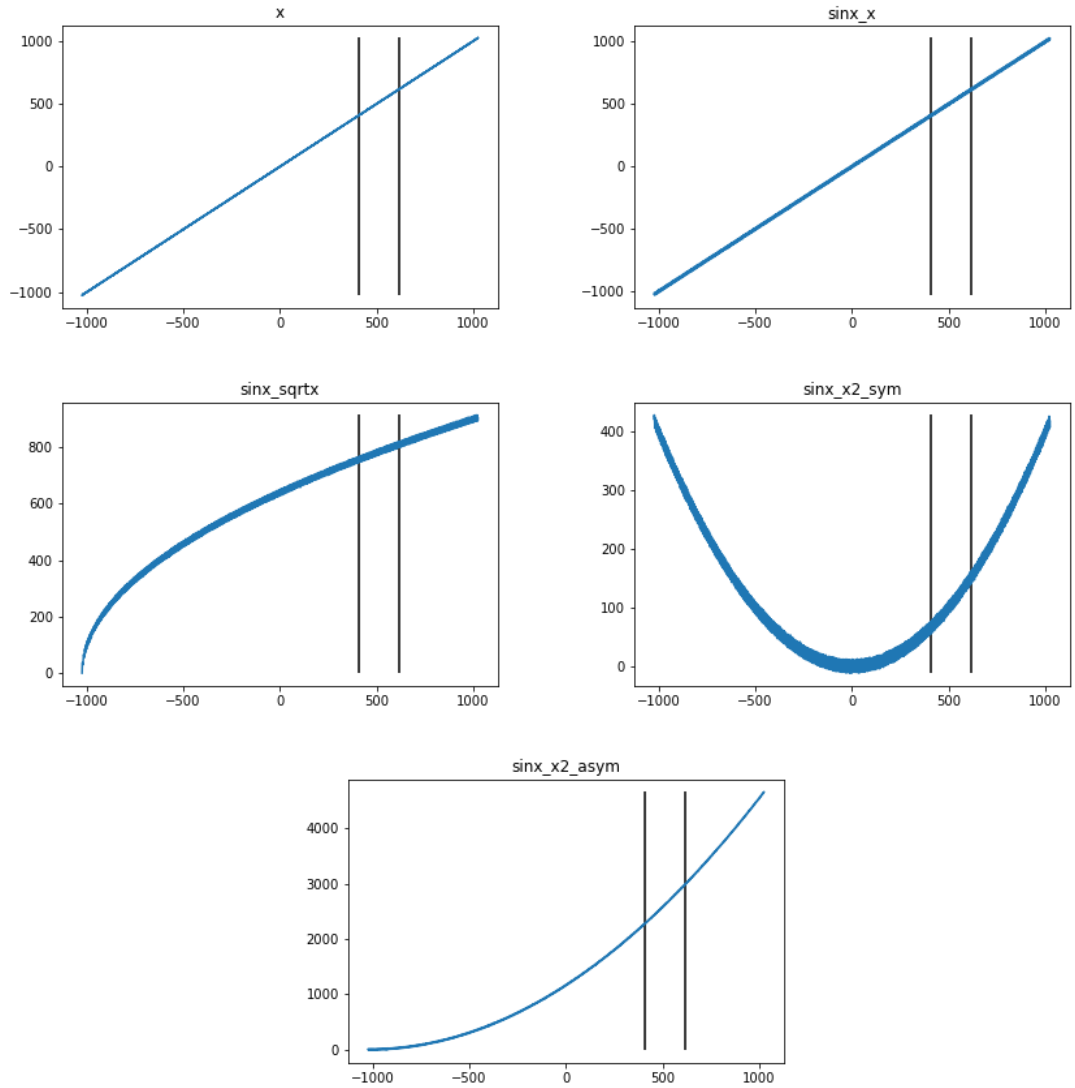

Figure 13: Synthetic datasets with underlying trend. The entire datasets are plotted, so the cyclical patterns by `sin` function is not visually identifiable. The black vertical lines split the the training, development and test datasets, from left to right.

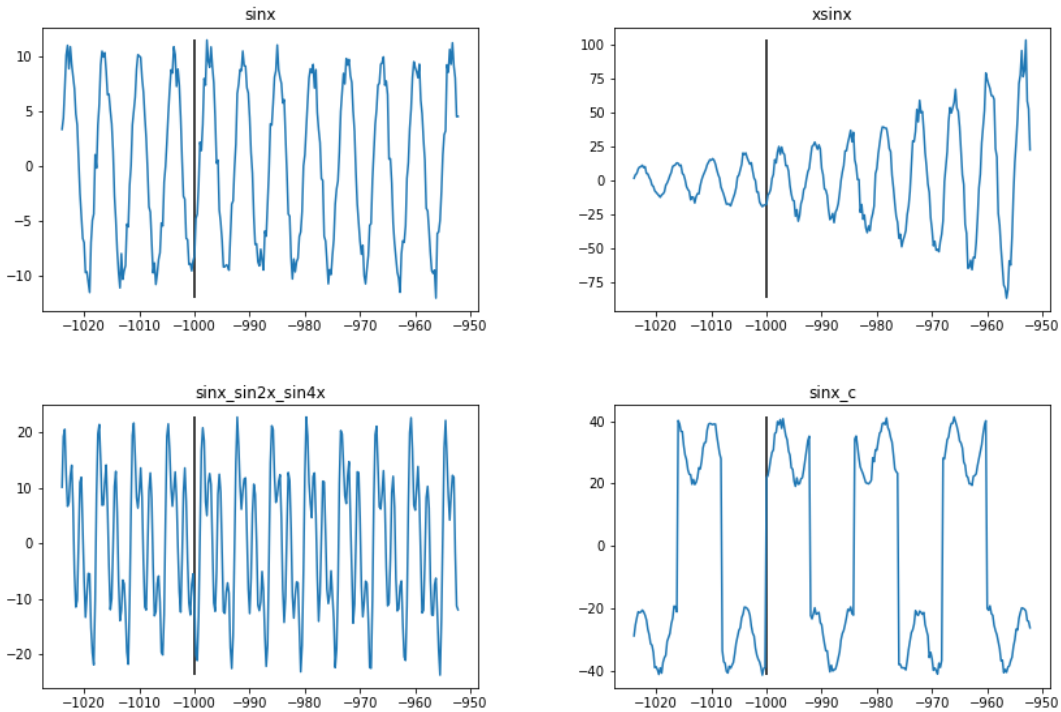

Figure 14: Synthetic datasets without underlying trend. Only one subsequence is plotted, where the black vertical line splits the encoder input and forecasting target.

B Prediction Plots

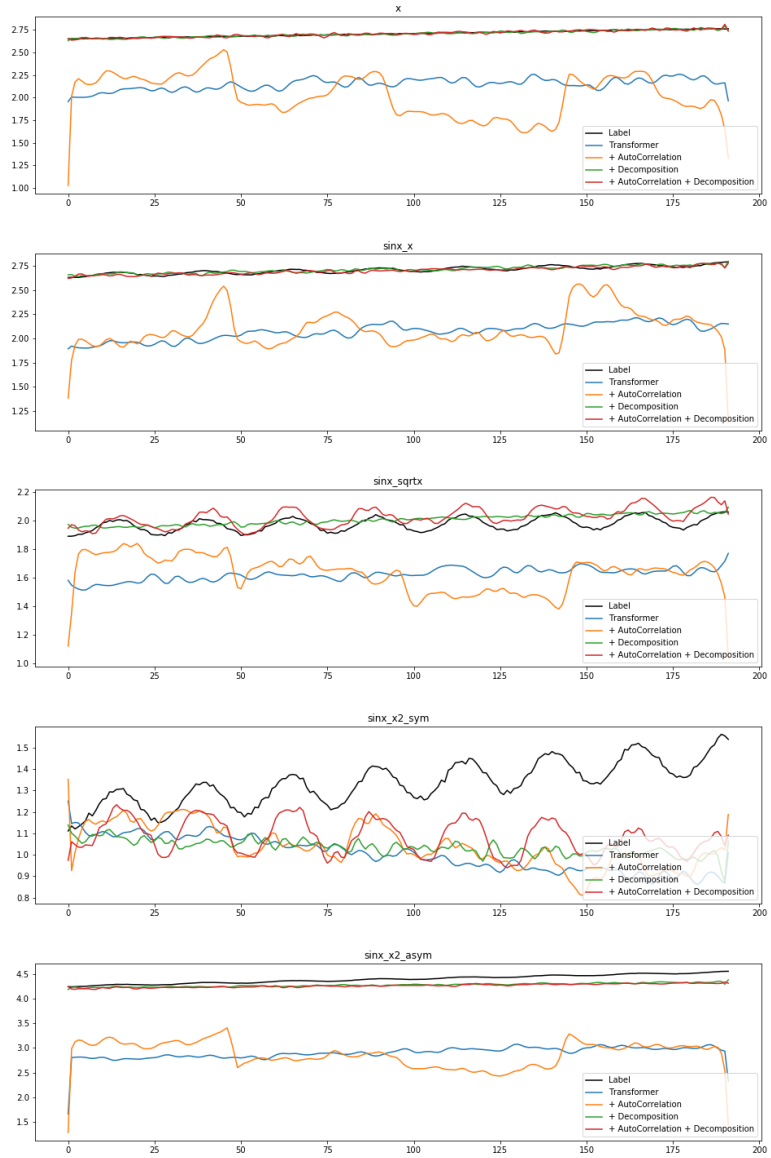

Figure 15: Forecasting one subsequence in synthetic datasets with underlying trend.

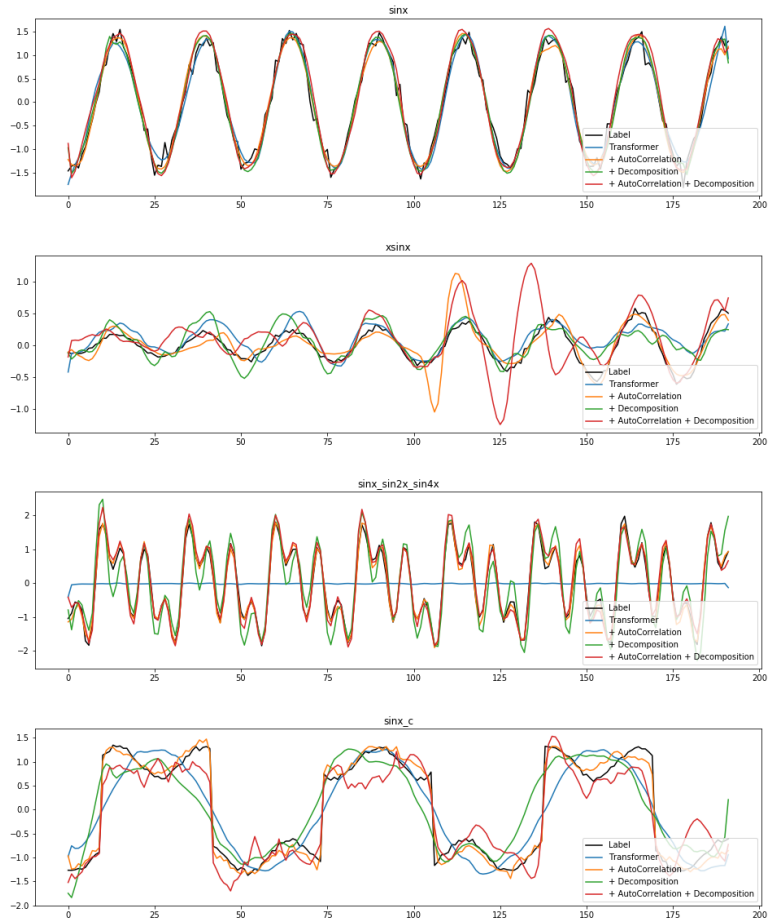

Figure 16: Forecasting one subsequence in synthetic datasets without underlying trend.

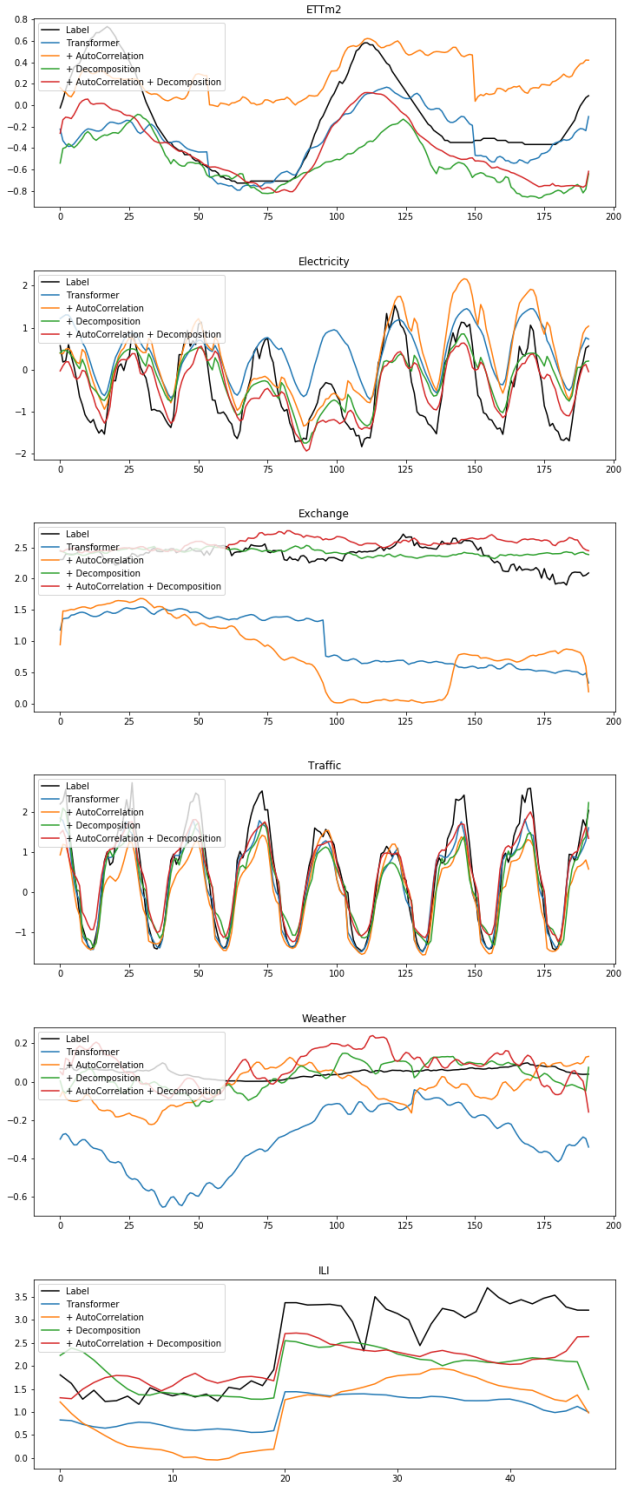

Figure 17: Forecasting one subsequence in synthetic datasets without underlying trend.

C AutoCorrelation Weights

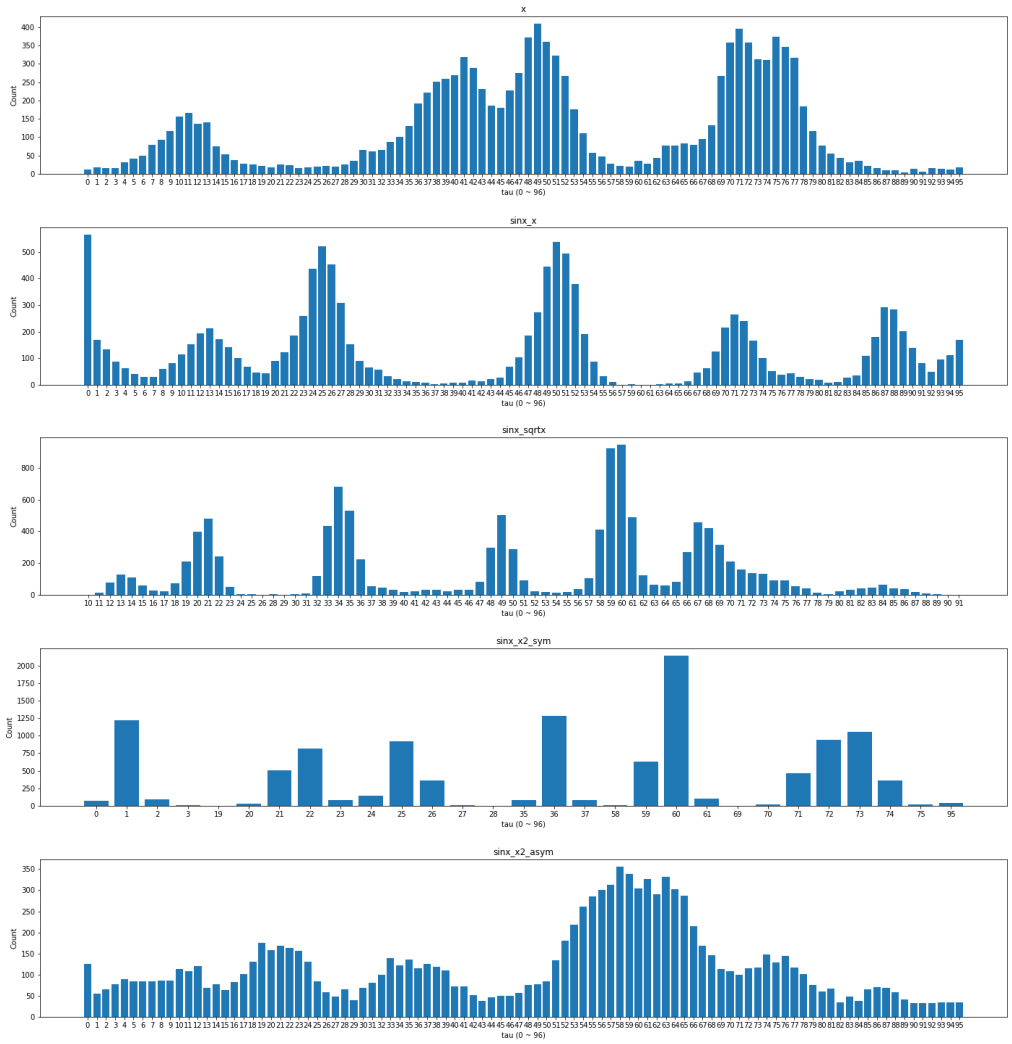

Figure 18: Distributions of τ on synthetic datasets with trend.

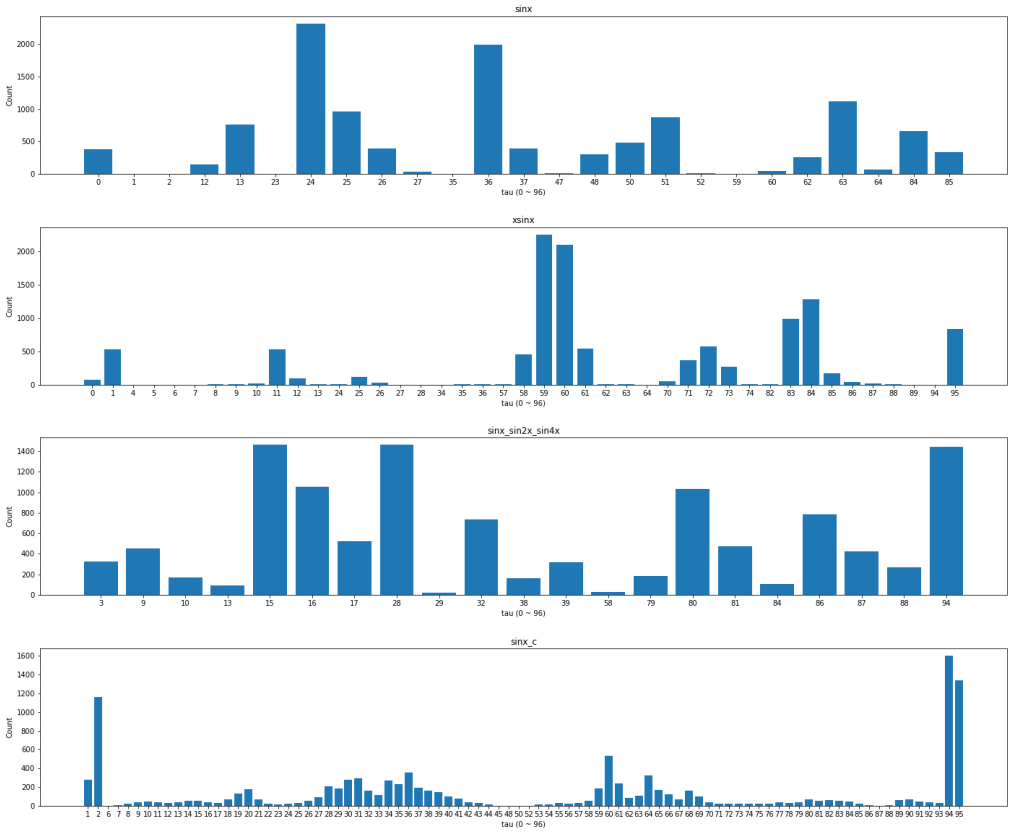

Figure 19: Distributions of τ on synthetic datasets without trend.