Are Transformers the Best Choice For Time Series Forecasting ?

Statistical Learning Final Project AY:2023-2024

SAPIENZA
UNIVERSITÀ DI ROMA

Gianluca Procopio (1942103)
Himel Ghosh (2102750)
Giorgio Bertone (2112729)
Filippo Parlanti (2136988)

Introduction

In our project we want to compare the performance of various models, from the simplest to the most advanced, to conduct Long Time Series Forecasting.

We want to answer the following questions:

- how well Transformers are able to solve this kind of problem?
- is it really necessary to bother the Transforms to do LTSF?
- how is their performance compared to that achieved by more simpler models?

Why do we “need” to study simpler models?

Let say it clearly: the more complex the models are, the more money is needed to train them. We encountered this problem precisely in trying to train the Transformer models. They took a long time just to train them with time horizon 96.

This is the reason why it is necessary to balance the *quality-price ratio* of the models to be chosen.

“I would like to live as a poor man with lots of money.” (Pablo Picasso)

Our “polar stars” of the project

- ***"Are Transformers Effective for Time Series Forecasting?"*** (https://arxiv.org/abs/2205.13504, 26/05/2022), Ailing Zeng, Muxi Chen, Lei Zhang, Qiang Xu

- ***"Frequency-domain MLPs are More Effective Learners in Time Series Forecasting"*** (https://arxiv.org/abs/2311.06184, 10/11/2023), Kun Yi, Qi Zhang, Wei Fan, Shoujin Wang, Pengyang Wang, Hui He, Defu Lian, Ning An, Longbing Cao, Zhendong Niu

Problem statement

We are dealing with a multivariate time series forecasting problem. In particular, let $X = [X_1, ..., X_T] \in \mathbb{R}^{N \times T}$ represent our regularly sampled multivariate time series, where:

- N is the number of distinct time series (variables)
- T is the total number of timestamps
- $X_t \in \mathbb{R}^N$ denotes the values of all N series at time t

Our models will work using a lookback window of length L. This means that the input matrix can be represented as $X_t = [X_{t-L+1}, ..., X_t] \in \mathbb{R}^{N \times L}$

Problem statement

We aim to forecast the next $\mathcal{T}$ timestamps for all N series, where the prediction target is defined as $Y_t = [X_{t+1}, ..., X_{t+\tau}] \in \mathbb{R}^{N \times \tau}$.

$\mathcal{T}$ represents the forecasting horizon, and in our project it will assume values in the set $\{96, 132, 336, 720\}$ as in the referenced articles.

For each model $f_\theta(\cdot)$, we will look towards minimizing the discrepancy between our predictions $\hat{Y}_t = f_\theta(\cdot)$ and the actual future values Y_t under the Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) losses.

The models we have tested

Statistical approaches:

- ARIMA
- Prophet

Machine Learning:

- GBR

Deep Learning:

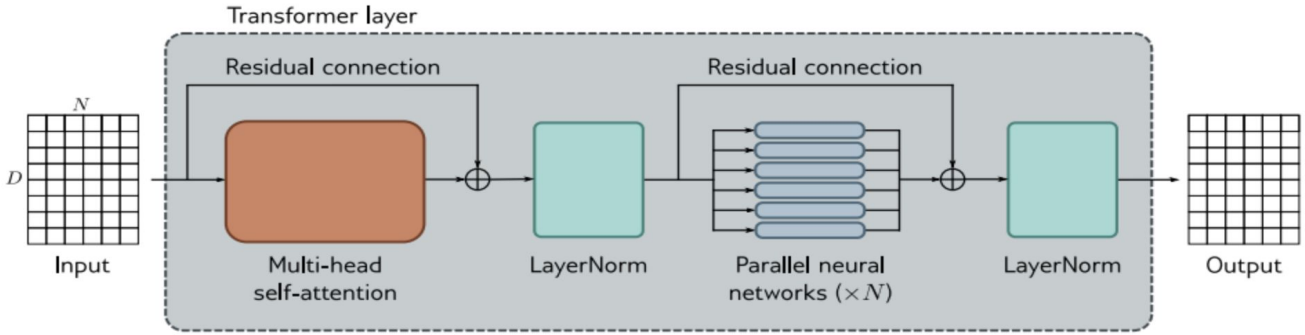
- MLP

Transformers:

- Informer
- Autoformer
- Vanilla Transformer
- PatchTST

Transformers:

- self-attention is permutation invariant
- it does not inherently preserve the order of the elements
- it is equivariant with respect to input permutations.
- positional encodings have been added to preserve some order.



Problem of Transformers in TS forecasting

Transformers fail to maintain the same order

Let's face it: The order of data points is crucial in Time Series, the self-attention mechanism in Transformer models leads to a loss of temporal information due to its permutation invariance.

Experiments show that **shuffling the order of time series data does not significantly impact the performance of Transformer models**, indicating their inadequacy in learning temporal dependencies.

LTSF-Linear models avoid this issue and perform more reliably on time series tasks.

Problem of Transformers in TS forecasting

Methods		Linear			FEDformer			Autoformer			Informer		
Predict Length		*Ori.*	*Shuf.*	*Half-Ex.*	*Ori.*	*Shuf.*	*Half-Ex.*	*Ori.*	*Shuf.*	*Half-Ex.*	*Ori.*	*Shuf.*	*Half-Ex.*
Exchange	96	0.080	0.133	0.169	0.161	0.160	0.162	0.152	0.158	0.160	0.952	1.004	0.959
	192	0.162	0.208	0.243	0.274	0.275	0.275	0.278	0.271	0.277	1.012	1.023	1.014
	336	0.286	0.320	0.345	0.439	0.439	0.439	0.435	0.430	0.435	1.177	1.181	1.177
	720	0.806	0.819	0.836	1.122	1.122	1.122	1.113	1.113	1.113	1.198	1.210	1.196
	Average Drop	N/A	27.26%	46.81%	N/A	-0.09%	0.20%	N/A	0.09%	1.12%	N/A	-0.12%	-0.18%
ETTh1	96	0.395	0.824	0.431	0.376	0.753	0.405	0.455	0.838	0.458	0.974	0.971	0.971
	192	0.447	0.824	0.471	0.419	0.730	0.436	0.486	0.774	0.491	1.233	1.232	1.231
	336	0.490	0.825	0.505	0.447	0.736	0.453	0.496	0.752	0.497	1.693	1.693	1.691
	720	0.520	0.846	0.528	0.468	0.720	0.470	0.525	0.696	0.524	2.720	2.716	2.715
	Average Drop	N/A	81.06%	4.78%	N/A	73.28%	3.44%	N/A	56.91%	0.46%	N/A	1.98%	0.18%

Table 5. The MSE comparisons of models when shuffling the raw input sequence. *Shuf.* randomly shuffles the input sequence. *Half-EX.* randomly exchanges the first half of the input sequences with the second half. Average Drop is the average performance drop under all forecasting lengths after shuffling. All results are the average test MSE of five runs.

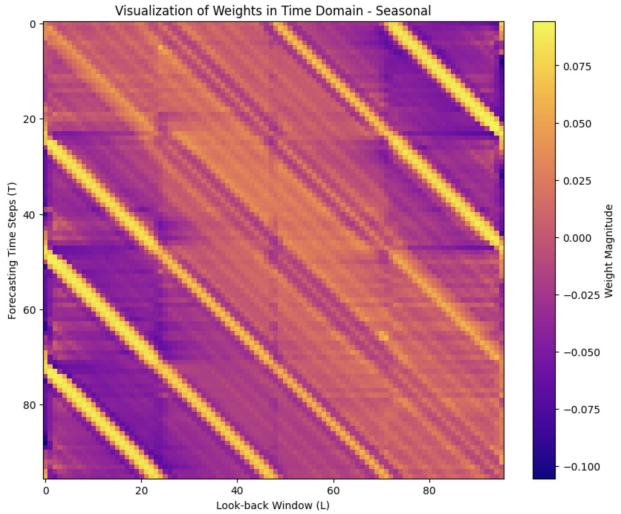
LTSF – Linear

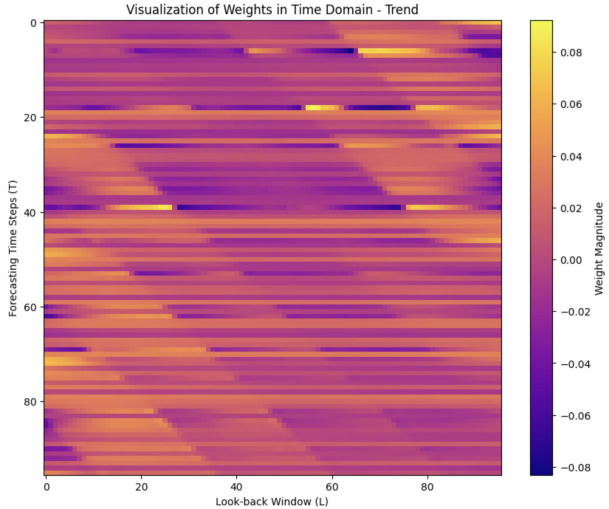
The authors introduce a family of linear models, which directly regress historical values for future predictions via a weighted sum operation

$$\hat{X}_i = WX_i$$

- **Vanilla Linear**: simple one-layer model
- **DLinear**: decompose the data into *trend* and *seasonality* component. Then apply two linear layers to both and sum up the two features to get the final prediction.
- **NLinear**: normalize the data before it goes through a linear layer and then denormalize the results.

LTSF Linear – DLinear





Plot shows there is a periodicity in the data, as the model gives high weights to the latest time step of the look-back window for the 0,23,47,.. forecasting steps with 24 steps being a day as we have hourly data.

LTSF Linear

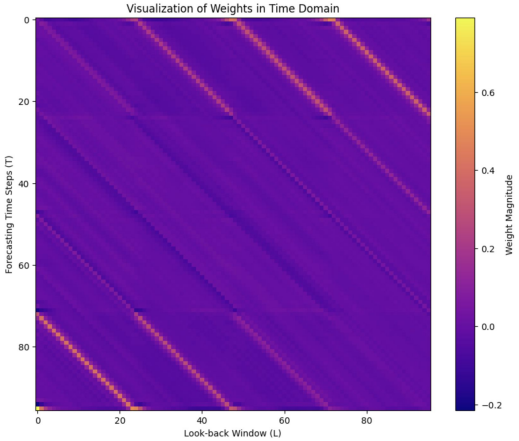
- Impact of input look-back window sizes is studied.
- **MAE** of different models for varying lookback window sizes in long term forecasting (T=96) on the **exchange rate** dataset is shown:

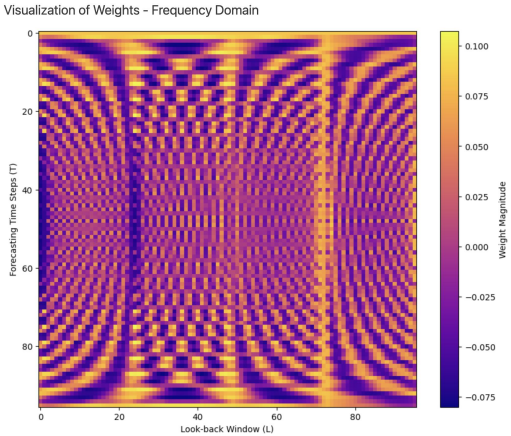
L	DLinear	NLinear	Informer	Autoformer	PatchTST	Vanilla Trans
24	0.4336	0.0952	0.1797	0.0650	0.0347	0.1911
48	0.3589	0.0739	0.1919	0.0601	0.0348	0.1902
96	0.1739	0.0599	0.1551	0.0653	0.0348	0.1501
192	0.0688	0.0471	N/A	N/A	0.0352	N/A

- The window size made no impact on model performance of transformers.
- Autoformer and PatchTST seem completely unaltered, Vanilla Transformer slightly improves, while Informer first gets worse and then improves
- LTSF-Linear models show clear improvement as the window size gets bigger.

Why MLPs in frequency domain?

- **MLP cannot handle global dependencies of time series**
- **Frequency Domain MLPs give a global view easier to learn spatial/temporal dependencies**
- **Frequency MLP offers Energy Compaction**
- **FreTS will learn Time Series forecasting mapping in Frequency domain.**

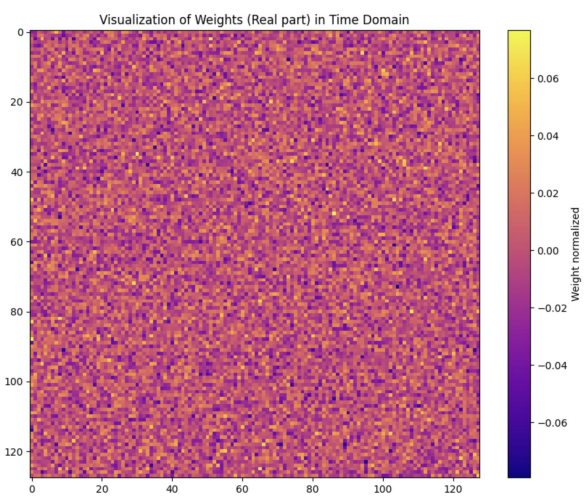


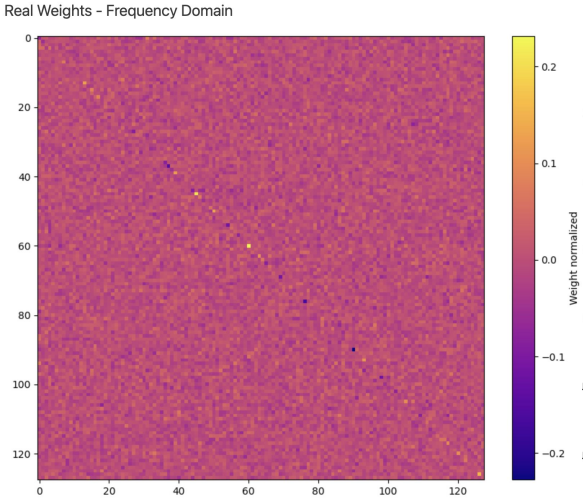


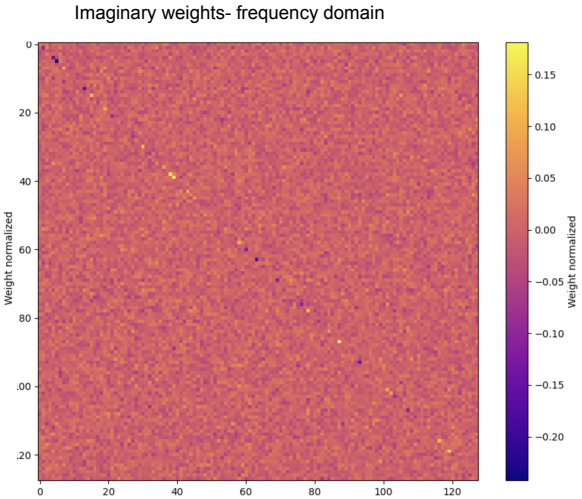
Patterns learned in the frequency domain exhibit more obvious global periodic patterns than those learned in the time domain

Why MLPs in frequency domain?

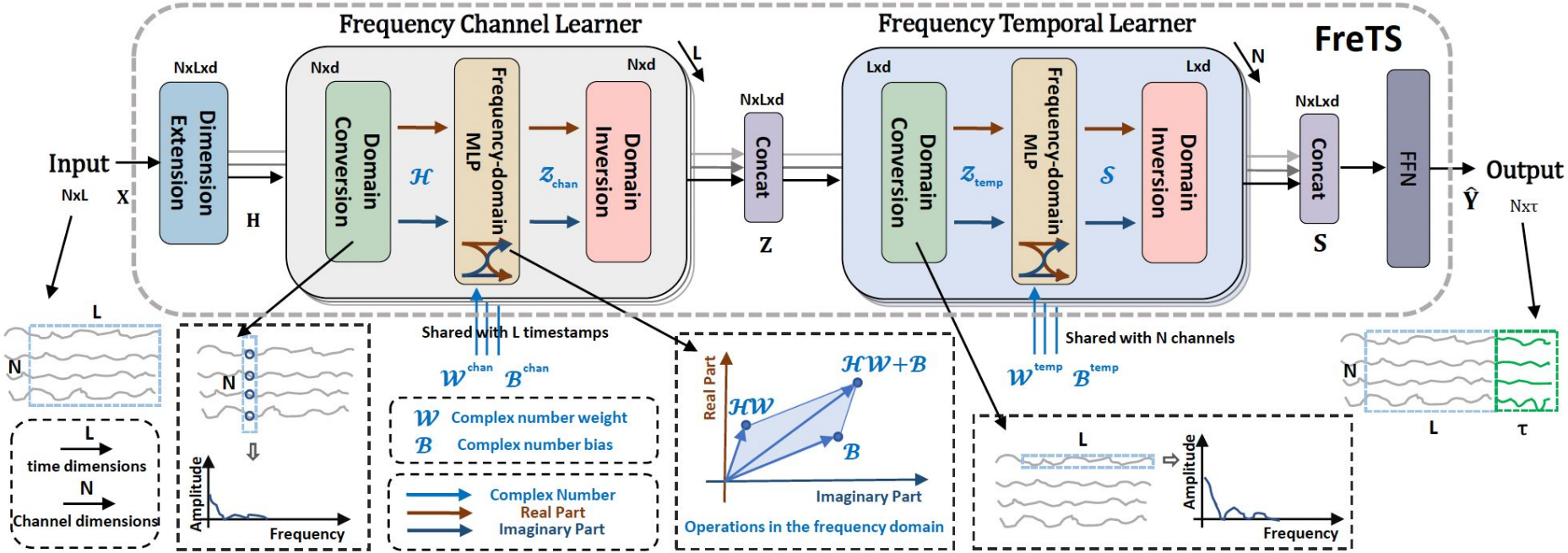
Energy Compaction: Learning in the frequency domain can identify clearer diagonal dependencies and key patterns than the time domain:







FreTS Architecture



$$HW+B=\mathcal{F}(h\circledast w+b)$$

Results

Data	Model	FreTS		GB Regression		ARIMA		Prophet		LTSF - Linear		Repeat	
	Metric	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
	Horizon												
Weather	96	0.041	0.092	0.049	0.055	0.087	0.108	**0.035**	**0.040**	0.089	0.120	**0.035**	0.044
	192	0.050	0.115	0.072	0.083	0.104	0.133	**0.038**	**0.044**	0.108	0.144	0.052	0.067
	336	0.062	0.164	0.074	0.085	0.142	0.191	**0.043**	**0.050**	0.143	0.184	0.051	0.069
	720	0.064	0.106	0.081	0.093	0.147	0.192	0.066	0.086	0.201	0.245	**0.055**	**0.075**
Exchange	96	0.050	0.068	0.036	0.043	**0.034**	0.040	0.085	0.089	0.044	0.060	0.042	**0.035**
	192	0.067	0.089	0.056	0.065	0.056	0.066	0.118	0.126	0.055	0.073	**0.051**	**0.060**
	336	0.102	0.130	0.082	0.098	**0.064**	**0.078**	0.162	0.176	0.080	0.105	0.081	0.098
	720	0.132	0.171	0.108	0.126	0.116	0.138	0.236	0.256	0.185	0.225	0.109	0.127
Traffic	96	**0.020**	**0.038**	0.106	0.127	0.087	0.118	0.042	0.064	0.041	0.079	0.094	0.131
	192	**0.021**	**0.040**	0.110	0.138	0.099	0.137	0.046	0.068	0.038	0.075	0.102	0.143
	336	**0.022**	**0.041**	0.111	0.142	0.098	0.136	0.047	0.072	0.039	0.076	0.099	0.141
	720	**0.023**	**0.043**	0.114	0.153	0.092	0.125	0.049	0.076	0.053	0.091	0.110	0.152
Electricity	96	**0.037**	**0.061**	0.195	0.223	0.163	0.191	0.086	0.106	0.054	0.080	0.145	0.172
	192	**0.038**	**0.063**	0.192	0.220	0.165	0.195	0.083	0.103	0.059	0.083	0.142	0.169
	336	**0.043**	**0.067**	0.187	0.214	0.168	0.199	0.085	0.104	0.056	0.082	0.147	0.175
	720	**0.051**	**0.078**	0.189	0.215	0.170	0.203	0.086	0.106	0.059	0.085	0.149	0.178
ETTh1	96	**0.063**	**0.091**	0.131	0.148	0.127	0.159	0.078	**0.091**	0.077	0.104	0.108	0.134
	192	**0.068**	0.095	0.142	0.161	0.126	0.156	0.075	**0.089**	0.097	0.125	0.121	0.150
	336	**0.072**	**0.099**	0.136	0.153	0.134	0.163	0.085	0.102	0.110	0.140	0.124	0.155
	720	**0.075**	**0.105**	0.142	0.162	0.159	0.192	0.102	0.118	0.133	0.167	0.144	0.178
ETTm1	96	**0.048**	**0.070**	0.111	0.135	0.142	0.174	0.073	0.090	0.089	0.120	0.091	0.114
	192	**0.055**	**0.081**	0.115	0.139	0.168	0.202	0.076	0.095	0.108	0.144	0.091	0.118
	336	**0.062**	**0.091**	0.121	0.143	0.199	0.234	0.081	0.101	0.143	0.184	0.102	0.131
	720	**0.070**	0.106	0.123	0.147	0.203	0.240	0.085	**0.105**	0.201	0.245	0.112	0.140

Conclusions

1. Transformers do not bring improvements in the field of Time Series forecasting. The increased complexity of these models does not translate into better performance.
2. Simpler architectures are better. LTSF-Linear, DLinear and NLinear and MLP models (in the frequency domain) showed better results. These models are way less complex than transformers and have higher performance in all the dataset that we have tested.
3. Classic and simple models like ARIMA, Gradient Boosting Regression and our baseline (Repeat) have shown overall poor performances. On the other hand, our Prophet implementation was able to keep up to almost all the best models we found.
4. If we admit a higher computational cost (compared to linear models) then we will definitely choose the MLP in frequency model to do time series prediction on long time horizons.