Transformers for time-series forecasting: a statistical approach

Cyril Garcia and Guillaume Remy

December 17, 2025

Abstract

In this paper, we propose a controlled synthetic benchmarking framework to evaluate transformer models for multivariate time-series forecasting under varying dependencies across time, series, and features, as well as under varying noise levels. We measure model performance using out-of-sample correlations to the known optimal ground-truth prediction. Two-way attention transformers — which alternate temporal and cross-sectional self-attention — outperform baselines (Lasso, boosting, MLPs) in a large number of cases, including in low signal-to-noise settings. Furthermore, we implement sparse attention, which yields some performance gains in the presence of noise. Future directions include real-world validation and the study of higher-order effects.

Contents

1 Introduction

Time-series forecasting is a cornerstone of data-driven decision-making across diverse domains, from finance and meteorology to supply chain and energy management. Traditional statistical models, such as ARIMA and exponential smoothing [1], have long dominated this field due to their interpretability and efficiency on univariate data. However, with the explosion of multivariate, high-dimensional datasets, these methods increasingly fall short in the presence of complex interdependencies across time and features. In recent years, transformer architectures—which originally revolutionized natural language processing through self-attention mechanisms [9]—have shown promise in time-series tasks and have led to numerous architecture proposals for forecasting, as reviewed in [10]. Notable examples include the Informer model for efficient long-sequence forecasting [13], Autoformer with decomposition and auto-correlation mechanisms [11], FEDformer leveraging frequency-enhanced attention [14], and PatchTST for patch-based representations [7]. Yet, recently, the effectiveness of transformers for time-series forecasting has been questioned, with works showing that simpler (linear) models can achieve comparable or superior performance [12, 3] on many real-world benchmark datasets.

To better understand the conditions under which transformer-based models actually outperform alternatives, we propose a controlled benchmarking framework using synthetic data, where both noise levels and types of dependencies in the forecasting problem can be varied, and the performance of a range of model choices can be statistically evaluated. More precisely, we frame the forecasting problem as predicting a target series $Y_{t,n}$ from predictors $X^{(j)}_{s,n}$, incorporating dimensions for time indexed by s, t (with $s \leq t$, ensuring causality), series indexed by n, and features indexed by j. To dissect model behaviors, we generate artificial datasets with tunable "orders" of effects: order 0 (simple linear dependencies), order 1 (shifts in time or cross-sections, or non-linear feature interactions), and order 2 (combined shifts). Noise is systematically varied, and we measure model performance using out-of-sample correlations against the optimal ground-truth dependency. A key assumption of this paper is that the time series of predictors X is stationary and has already been preprocessed and normalized. This allows us to focus solely on evaluating the transformers' ability to capture dependencies at different noise levels, leaving the preprocessing stage for a later study.

Our contributions are threefold. First, we test a series of transformer architectures with one-way and two-way attention mechanisms—two-way referring to applying attention along both the time-series and cross-sectional dimensions (as in [5, 6])—to exploit the multi-dimensional structure of time-series data. We benchmark these models against baselines like Lasso regression [8], boosting [4], as well as simpler neural network architectures, revealing that transformers—particularly two-way models—outperform traditional methods on a number of effects, even at high noise levels. Second, we propose and test a sparse attention implementation [2] to enhance robustness in low-signal regimes, which yields up to 10–20% performance gains. Third, we provide in the appendix an analytical computation that gives the expected correlation in the linear case, bridging empirical results with statistical theory.

The remainder of the paper is organized as follows: Section 2 details the problem setup, data generation, models, and sparse attention. Section 3 presents experimental results, including noise-level comparisons and sparsity tests. Section 4 concludes with implications and future directions, such as scaling to real datasets and higher-order effects.

2 Setup

2.1 The forecasting problem

We start by considering the following general time-series forecasting problem of the target series Y using the time-series of predictors X:

$$Y_{t,n} = \mathcal{F}\left(\left(X^{(j)}_{s,n}\right)_{s\leq t,\ n\leq N,\ j\leq F}\right) + \epsilon. \tag{1}$$

Here the indices s, t and n denote respectively the time and series indices, they obey $1 \leq s \leq t \leq T$ and $1 \leq n \leq N$. We will refer to them frequently as the time-series and cross-sectional directions. The series X also contains a features dimension indexed by j satisfying $1 \leq j \leq F$. The quantity $Y_{t,n}$ can depend a priori on an arbitrary function $\mathcal{F}$ of all the $X^{(j)}_{s,n}$ plus some noise ϵ, the only constraint being $s \leq t$ implying the temporal dependence can only be up to the present time t. In this paper we also assume stationarity of the series, namely that the functional dependence $\mathcal{F}$ is invariant by shifts $t \to t + t_0$, $t_0 \geq 0$ (see below the concrete examples of $\mathcal{F}$).

Remark 1. *To give an example of what the T, N, F dimensions could correspond to, assume a forecasting problem where one is interested in predicting the temperature in different cities on different days. For this it is natural to set:*

- ***T**: The time index t indexes the days of the year.*
- ***N**: The series index n corresponds to the different cities or locations.*
- ***F**: The feature index j corresponds to the different available features for forecasting, which could be for instance temperature, wind, humidity, etc...*

$X^{(j)}_{t,n}$ then contains all the data above and $Y_{t,n}$ would correspond to the one day forward temperature we wish to predict. In this setup if $j = 1$ corresponds to the temperature feature one would have $Y_{t,n} = X^{(1)}_{t+1,n}$ but in general this doesn't need to be the case. In some setups it is also possible one only has a single feature, i.e. $F = 1$, or that the target time-series Y is only one-dimensional, i.e. $N = 1$, but we keep all three dimensions to treat the most general case.

To make progress in understanding this forecasting problem we now make some assumptions on the function $\mathcal{F}$. The simplest case, which we will call the *Order 0*, corresponds to assuming the linear dependence

$$Y_{t,n} = \sum_j \rho_{j,n} X_{t,n}^{(j)} + \epsilon, \tag{2}$$

for $\rho_{j,n} \in [0,1]$ and for every t, n. In practice $\rho_{j,n}$ could be independent of n if the effect is assumed to be the same for all series.

Beyond this simple linear case we can consider shifted relationships, either in the time-series or cross-section directions, namely

$$Y_{t,n} = \sum_j \rho_{j,n} X_{t-s_{j,n},n}^{(j)} + \epsilon, \tag{3}$$

and

$$Y_{t,n} = \sum_j \rho_{j,n} X_{t,n+k_{j,n}}^{(j)} + \epsilon, \tag{4}$$

where in the first case the shifts $s_{j,n} \geq 0$ need to be non-negative, and in the second case the sum $n+k_{j,n}$ should be understood modulo the number N of series. Alternatively we can also replace the linear sum of (2) by a function $G : \mathbb{R}^F \to \mathbb{R}$ of the features, for $\rho \in [0,1]$:

$$Y_{t,n} = \rho\, G(X_{t,n}^{(1)}, \ldots, X_{t,n}^{(F)}) + \epsilon. \tag{5}$$

As a very basic example of G we could take $X_{t,n}^{(j_1)} \mathrm{sign}(X_{t,n}^{(j_2)})$, the conditioning of feature j_1 by the sign of feature j_2. Another simple case is any non-linear function of just one of the features, i.e. $G(X_{t,n}^{(j_1)})$. We call (3), (4), and (5) the *Order 1* effects. In this paper we will mostly test the models on the order 0 and 1 effects above but in the same spirit we also consider the double shift relationship

$$Y_{t,n} = \sum_j \rho_{j,n} X_{t-s_{j,n},n+k_{j,n}}^{(j)} + \epsilon, \tag{6}$$

which we classify as an *Order 2* effect.

2.2 Generating artificial data

Let us now explain our procedure to generate artificial data for the series X and Y. We proceed with the following steps.

- Sample $X_{t,n}^{(j)}$ i.i.d. $\mathcal{N}(0,1)$ for all indices t, n, j.
- Combine the $X_{t,n}^{(j)}$ following one or a sum of the effects and obtain an $\widetilde{Y}_{t,n}$, the $Y_{t,n}$ without the noise. For instance for the shift effect (3) one would define $\widetilde{Y}_{t,n} := \sum_j \rho_{j,n} X_{t-s_{j,n},n}^{(j)}$. Note $\widetilde{Y}_{t,n}$ is the optimal prediction of the model.
- Assuming $\widetilde{Y}_{t,n}$ has been constructed to have mean 0 and variance $\rho_n^2 < 1$, define $Y_{t,n} = \widetilde{Y}_{t,n} + \sqrt{1-\rho_n^2} Z_{n,t}$, with $Z_{n,t}$ i.i.d. $\mathcal{N}(0,1)$.

A model $\mathcal{M}$ will then, given $X_{s,n}^{(j)}$ for all n, j and $s \leq t$, output $\widehat{Y}_{t,n}$, a prediction for $Y_{t,n}$. We can then measure the performance of the model $\mathcal{M}$ by computing the correlation between $\widehat{Y}_{t,n}$ and $Y_{t,n}$ or $\widetilde{Y}_{t,n}$.

2.3 Models and architectures for forecasting

In this section we give the list of models and architectures we will be testing to predict the different effects of the synthetic data. We introduce here a parameter T_r, the r subscript standing for rolling window. This corresponds to the furthest lags of X we will use to forecast Y, namely we will only use $X_{s,n}^{(j)}$ for $t - T_r < s \leq t$ to predict $Y_{t,n}$.

i) Lasso regression. As the most basic benchmark to predict Y using X we write down the linear regression:

$$Y_{t,n} = \sum_{s=0}^{T_r-1} \sum_{n',j} \beta_{s,n',j} X_{t-s,n'}^{(j)} + \epsilon. \tag{7}$$

Note that this model assumes there is no time-series structure and treats every entry of X as an independent feature for prediction, using the time-series window of length T_r. We will then estimate the coefficient $\beta_{s,n',j}$ using Lasso regression where the λ penalty parameter is estimated using the standard cross-validation.

ii) Global multi-layer perceptron. Next we move to a multi-layer perceptron (MLP) but which also assumes no time-series structure, which we call the global MLP. This model flattens the input $X_{s,n}^{(j)} \in \mathbb{R}^{T_r \times N \times F}$ into a vector of dimension $T_r \times N \times F$ and applies four 512-dimensional linear layers with GELU activations and 0.1 dropout, followed by a final projection to N outputs. This model gives a simple neural network baseline without making any use of the time-series structure.

iii) Two-way multi-layer perceptron. Next we consider a model that will make explicit use of the multi-dimensional time-series structure. We implement a *two-way* MLP that separately processes the input along the time-series and cross-sectional dimensions. Given an input tensor $\mathbf{X} \in \mathbb{R}^{B \times T_r \times N \times F}$ corresponding to B batches of $X_{s,n}^{(j)}$, we consider two processing orders:

Time first ("T"):

1. Reshape and permute $\mathbf{X}$ to $(B \cdot N, T_r \cdot F)$.
2. Apply a deep MLP with $L_1 = 3$ layers of width $h_1 = 256$, GELU activations, dropout ($p = 0.1$), yielding an output tensor $\mathbf{E} \in \mathbb{R}^{B \cdot N \times h_1}$.
3. Reshape $\mathbf{E}$ to have dimensions $(B, N \cdot h_1)$ and feed into a second MLP ($L_2 = 3$ layers, width $h_2 = 512$) that outputs the final predictions $\hat{\mathbf{Y}} \in \mathbb{R}^{B \times N}$.

Cross-section first ("C"): The operations are symmetric: first process along the variable dimension $(B \cdot T_r, N \cdot F) \to (B \cdot T_r, h_1)$, then reshape and feed into a second MLP $(B, T_r \cdot h_1) \to (B, N)$ to get the final prediction.

iv) Transformers with one-way attention. We now arrive at our first transformer model which will apply an attention mechanism in one of the two directions of the problem, the other direction being compressed by a simple linear layer or an MLP. Hence there are again two choices "T" and "C" based on which direction we apply the attention along.

- **Temporal attention ("T")**: The input is reshaped to $(B, T_r, N \cdot F)$ and projected to dimension $(B, T_r, d_{\text{model}})$ with $d_{\text{model}} = 256$ via either a single linear layer or a deep MLP compressor ($L_c = 4$ layers). Learned positional embeddings are added along the time axis. A standard Transformer encoder ($L = 2$ layers, $H = 8$ heads, FFN dim $= 512$, dropout $= 0.2$) processes the sequence. At the end we project the d_{model} to N from which we obtain a (B, T_r, N) tensor from which we can select the last time step as prediction.

- **Cross-sectional attention ("C")**: Symmetric to the above: input is this time reshaped to $(B, N, T_r \cdot F)$ and is compressed via a linear layer or MLP to a (B, N, d_{model}) tensor. We process this input with the same Transformer as above with attention along the N dimension. At the end we project the d_{model} dimension to 1 and obtain the (B, N) prediction.

v) Transformers with two-way attention. As our most general transformer architecture, we propose a model which applies self-attention along the time-series and cross-sectional dimensions following a custom ordering of time-series and cross-sectional attention blocks. Given again an input $\mathbf{X} \in \mathbb{R}^{B\times T_r \times N \times F}$, the architecture proceeds as follows:

1. The feature dimension F is linearly projected to $d_{\text{model}} = 256$.
2. Two learned positional embeddings are added: one for time steps and one for the N series.
3. The core consists of L attention blocks defined by a string (e.g., `"TCTC"`):
   - **T-block**: Independent temporal attention over the T_r dimension for each of the N series (shape: $B \cdot N \times T_r \times d_{\text{model}}$).
   - **C-block**: Independent cross-sectional attention over the N variables at each time step (shape: $B \cdot T_r \times N \times d_{\text{model}}$).

   Each attention output is added to its input via a residual connection, followed by LayerNorm and dropout ($p = 0.2$).
4. An output head (LayerNorm $\to$ GELU $\to$ Linear) produces the final prediction for all N targets.

vi) Boosting model. Finally as a simple non-linear ML benchmark we will fit a standard boosting model on the predictors $X_{s,n}^{(j)}$, again with the time s constrained to $t - T_r < s \leq t$, to predict the targets $Y_{t,n}$. This gives along with Lasso a standard benchmark but which has the possibility of capturing non-linear effects.

2.4 Implementing sparse attention for transformers

To improve the robustness of our transformer architectures in low signal-to-noise regimes we introduce a sparse attention mechanism within the scaled dot-product attention computation. This approach dynamically prunes the attention matrix by computing the positions of k largest attention weights per row averaged over batches.

More formally, given query Q, key K, and value V tensors, we first compute the attention logits as $A = \frac{QK^\top}{\sqrt{d}} + b$, where d is the head dimension and b is an initial bias tensor incorporating any provided attention mask (e.g., causal mask where $b_{i,j} = -\infty$ for $j > i$) or set to zero otherwise. Next we apply a softmax function and compute an average over batches to get an average probability $\overline{P}$. Then for each row in $\bar{P}$ (corresponding to a specific head and query position), we apply a sparsification function to retain only the top-k entries along the key dimension, yielding a binary mask m. In our experiments we set $k = 3$. This mask is then used to create a new bias tensor b', initialized to zero and updated such that $b' = -\infty$ where $m = 0$. We add this to the original logits: $A' = A + b'$ (with broadcasting over the batch dimension). The final attention weights are computed by applying a softmax to A', which we multiply to the value tensor V.

This sparsification, applied per head and query position based on batch-averaged probabilities, effectively filters out weak or noisy attention links, promoting focus on high-confidence dependencies. It is integrated into our one-way and two-way transformer models, and empirical results in Section 3.2 demonstrate its benefits in high-noise settings.

3 Experiments and results

We will now present our experiments on synthetic data and the results. For the experiments below we set the global parameters of the data to be: $T = 5000$, $N = 10$, $F = 20$, $T_r = 5$. We construct the X and Y based on the procedure outlined in Section 2, using the effects given by equations (2) - (6). In each of these equations we will assume the ρ_j is independent of n, and we will choose it to be equal to a reference value ρ with probability 1/2 and equal to 0 otherwise. In the results table below we call the effects generated from equations (2), (3), (4), (5), and (6) to be respectively the linear, conditional, TS shift, CS shift, and TS-CS shift effects. The correlation level corresponds to the global parameter ρ.

We split the $T = 5000$ into 3500 points for training the model and 1500 for prediction. For each model we will measure its performance by reporting the out-of-sample correlation between its prediction $\widehat{Y}_{t,n}$ and the optimal model prediction $\widetilde{Y}_{t,n}$. In the tables below we will consider the following models that have been described in detail in Section 2.3:

- Lasso with cross-validation and boosting as simple linear and non-linear benchmarks.
- Global MLP and 2D MLP correspond respectively to models ii) and iii) of the models section, with the two-way MLP iii) being used in the time first ("T") configuration.
- We then have three transformer models, 1D Trans corresponding to model iv) in the temporal attention ("T") configuration, and $(TC)^2$ Trans and $(TC)^4$ Trans corresponding respectively to the transformers with two-way attention of v) respectively with layers `"TCTC"` and `"TCTCTCTC"`).

In the next subsections we will test all these models on all the effects at different levels of correlation ρ.

3.1 Comparison of all models at different noise levels

We present a first series of results where each table below corresponds to the performance of all the models at all levels of correlation ρ on one of the effects. First the results for the linear effect (2):

Correl level	Lasso	Boosting	Global MLP	2D MLP	1D Trans	$(TC)^2$ Trans	$(TC)^4$ Trans
0.01	0.016	-0.007	0.054	0.037	0.046	0.166	0.131
0.03	0.014	-0.005	0.158	0.113	0.165	0.160	0.106
0.05	0.042	0.006	0.216	0.202	0.260	0.492	0.364
0.1	0.114	0.034	0.432	0.454	0.478	0.740	0.639
0.3	0.304	0.181	0.817	0.851	0.840	0.974	0.982

Next the conditional effect (5):

Correl level	Lasso	Boosting	Global MLP	2D MLP	1D Trans	$(TC)^2$ Trans	$(TC)^4$ Trans
0.01	-0.000	-0.005	0.000	-0.008	-0.002	0.091	0.095
0.03	-0.003	0.003	-0.000	0.001	0.005	0.081	0.080
0.05	0.006	0.020	0.007	0.021	0.016	0.320	0.276
0.1	0.000	0.005	0.005	0.017	0.024	0.449	0.535
0.3	0.001	-0.012	-0.014	0.054	0.073	0.785	0.726

Next the time-series shift effect (3):

Correl level	Lasso	Boosting	Global MLP	2D MLP	1D Trans	$(TC)^2$ Trans	$(TC)^4$ Trans
0.01	0.014	0.020	0.050	0.025	0.009	0.140	0.102
0.03	0.030	0.006	0.136	0.104	0.032	0.128	0.109
0.05	0.056	0.025	0.214	0.196	0.135	0.489	0.560
0.1	0.160	0.046	0.410	0.457	0.316	0.688	0.658
0.3	0.310	0.175	0.816	0.854	0.799	0.981	0.971

Next the cross-sectional shift effect (4):

Correl level	Lasso	Boosting	Global MLP	2D MLP	1D Trans	$(TC)^2$ Trans	$(TC)^4$ Trans
0.01	0.018	0.012	0.048	0.045	0.047	0.037	0.036
0.03	0.006	-0.001	0.124	0.091	0.148	0.113	0.133
0.05	0.051	0.013	0.226	0.217	0.266	0.511	0.463
0.1	0.135	0.040	0.434	0.478	0.493	0.761	0.647
0.3	0.302	0.179	0.822	0.865	0.842	0.973	0.965

Finally the time-series cross-sectional double shift effect (6):

Correl level	Lasso	Boosting	Global MLP	2D MLP	1D Trans	$(TC)^2$ Trans	$(TC)^4$ Trans
0.01	0.003	0.006	0.059	0.035	0.015	0.015	0.010
0.03	0.044	0.010	0.146	0.096	0.038	0.009	0.003
0.05	0.036	0.006	0.235	0.226	0.129	0.024	0.012
0.1	0.149	0.044	0.424	0.460	0.329	0.036	0.135
0.3	0.314	0.187	0.813	0.861	0.803	0.974	0.974

From the tables above we can make the following observations:

- Linear effect: The two-way transformer models work extremely well, even with very low correlation.
- Conditional effect: Only the two-way transformer models succeed at finding this effect, all other models output noise out-of-sample in this case.
- TS shift effect: The two-way transformer is the best, one-way not nearly as good at low correlation. The MLPs have good performance.
- CS shift effect: Two-way transformers perform overall best, but the one-way transformer and MLPs are very close and even do slightly better for the 1% correlation.
- TS-CS shift effect: This is the one effect where the two-way transformers fail to find the effect (apart from at the 30% correlation) while the MLP models, especially the global one, perform very well.

Overall these results demonstrate a very strong performance of the two-way transformer models relative to all the other models on all the effects except for the double CS-TS shift effect.

For the correlation level of 0.1 we also perform a test combining all the effects at once to test how well the previous results on each model hold when all the effects are present at once. The synthetic data X is generated following the exact same procedure except that we split the features in five groups, where in each group the features correspond to one of the five effects. Instead of just having the optimal model prediction $\widetilde{Y}$, we here can define a $\widetilde{Y}$ per effect corresponding to just adding the features of X with the corresponding effect. Accordingly in the table below the columns correspond to the correlation with a different target: optimal represents the correlation to $\widetilde{Y}$, true the correlation to Y (i.e. $\widetilde{Y}$ plus the noise), and the other columns are the correlations to the specific effects.

	True	Optimal	Linear	Conditional	TS Shift	CS Shift	TS CS Shift
Lasso	0.041	0.121	0.099	0.010	0.068	0.083	0.023
Boosting	0.023	0.038	0.030	0.004	0.012	0.028	0.015
Global MLP	0.106	0.337	0.203	0.009	0.191	0.203	0.178
2D MLP	0.113	0.360	0.218	0.020	0.226	0.201	0.173
1D Trans	0.096	0.321	0.250	0.017	0.119	0.255	0.109
$(TC)^2$ Trans	0.197	0.631	0.354	0.215	0.333	0.348	0.222
$(TC)^4$ Trans	0.195	0.624	0.323	0.240	0.337	0.377	0.178

The above table clearly demonstrates the two-way transformer model is the best overall (highest correlation to the Optimal prediction) and performs pretty evenly across all the effects.

3.2 Testing the performance of sparse attention

Next in this section we present the results of using the sparse attention procedure described in Section 2.4. We test this procedure on the $(TC)^2$ Trans model of the previous subsection, which was the model with the best overall performance. Below we give two tables, each showing the performance of the model on every effect at every correlation level, with the first table being without sparsity and the second one with. The last column, named All, corresponds to combining all the effects at once just as in the very last table above.

First the $(TC)^2$ Trans model without sparsity:

Correl level	Linear	Conditional	TS Shift	CS Shift	TS-CS Shift	All
0.01	0.183	0.066	0.120	0.052	-0.000	0.091
0.03	0.142	0.088	0.129	0.107	0.001	0.107
0.05	0.552	0.317	0.410	0.563	0.021	0.358
0.1	0.704	0.410	0.723	0.666	0.500	0.592
0.3	0.978	0.773	0.979	0.976	0.977	0.932

Second the $(TC)^2$ Trans model with sparsity:

Correl level	Linear	Conditional	TS Shift	CS Shift	TS-CS Shift	All
0.01	0.175	0.076	0.150	0.042	0.006	0.092
0.03	0.152	0.082	0.115	0.135	0.008	0.112
0.05	0.609	0.261	0.402	0.493	0.028	0.300
0.1	0.781	0.442	0.670	0.749	0.458	0.617
0.3	0.980	0.747	0.971	0.973	0.972	0.931

Above we see that sparse attention enhanced the two-way transformer's robustness in low-signal regimes, yielding improvements in cases like conditional effects at 0.01 (0.076 with sparsity vs. 0.066 without) and TS shifts (0.150 vs. 0.120), though results are mixed for linear effects.

4 Conclusion and outlooks

In this paper, we have presented a statistical framework for evaluating transformer architectures in multivariate time-series forecasting, focusing on their ability to capture temporal and cross-sectional dependencies in synthetic data with controlled effects and noise levels. Our experiments demonstrate that two-way transformer models, which alternate self-attention along time-series ("T") and cross-sectional ("C") dimensions, consistently outperform traditional baselines such as Lasso regression and boosting, as well as neural alternatives like global and two-way MLPs, and one-way transformers. This superiority is particularly evident in high-noise regimes (e.g., correlation levels of 0.01), where $(TC)^2$ and $(TC)^4$ variants achieve correlations to the optimal prediction that are 5-10 times higher than Lasso for linear, conditional, and shift effects. When all effects are superimposed at a 0.1 correlation level, two-way transformers capture multifaceted dependencies with correlations up to 0.631, far surpassing Lasso's 0.041, highlighting their robustness in realistic, noisy scenarios.

Furthermore, the incorporation of sparse attention proves beneficial for enhancing performance in low-signal environments, yielding modest gains (e.g., 10-25% relative improvement) for conditional and TS shift effects at 0.01 correlation, though results are less pronounced for linear cases. This suggests that sparsity aids in filtering irrelevant dependencies, mitigating overfitting in data-scarce or noisy conditions. Hyperparameter choices, such as attention block ordering and depth, show sensitivity to noise: deeper $(TC)^4$ models perform better at moderate signals, while shallower variants suffice at extremes. The appendix's analytical computation provides a theoretical baseline for linear correlations, aligning with empirical observations that finite-sample effects diminish predictive accuracy at small T.

Finally we propose several avenues for future work. First, applying these architectures to real-world datasets—such as financial time-series, climate records, or sensor networks—could validate their efficacy beyond synthetics, potentially incorporating domain-specific priors like seasonality or exogenous variables. Second, investigating higher-order effects (e.g., interactions across multiple features or non-stationary dependencies) may reveal limitations or necessitate architectural innovations, such as adaptive sparsity or hybrid models integrating transformers with probabilistic components. Third, scalability tests with increased dimensions (e.g., larger T, N, or F) are crucial to assess computational efficiency and generalization, possibly leveraging distributed training or efficient attention variants like Performer or Reformer. By advancing transformers through this statistical lens, we aim to foster more interpretable and reliable forecasting tools for complex, noisy data.

5 Appendix: Analytical derivation in the linear case

To get a sense of the level of correlation between our model prediction $\widehat{Y}$ and the optimal prediction $\widetilde{Y}$ we expect to find as a function of the size of the data (total time steps, number of series and features, lags used), let's perform an analytic computation in the linear case where there is no time-series structure (i.e. flatten all features).

Let $X_1, \ldots, X_N$ be N i.i.d. $\mathcal{N}(0,1)$ random variables. Consider $\widetilde{Y} := \sum_{i=1}^{N} \rho_i X_i$ with $\rho_i \geq 0$ and $\rho^2 := \sum_{i=1}^{N} \rho_i^2 < 1$. Define then $Y = \widetilde{Y} + \sqrt{1-\rho^2} Z$ with $Z \sim \mathcal{N}(0,1)$ independent of all the X_i. Consider now T i.i.d. samples of $Y, X_1, \ldots, X_N$ and the multidimensional regression problem:

$$Y_t = \sum_i \beta_i X_{t,i} + \epsilon.$$

We can then estimate the β_i using OLS or Ridge

$$\widehat{\beta} = (X^T X + \lambda Id)^{-1} X^T Y,$$

and from here we obtain the model prediction $\widehat{Y} = X\widehat{\beta}$. The goal is thus to understand the distribution of $Corr(\widehat{Y}, \widetilde{Y})$ as a function of N, T. To start we will just compute the expectation.

i) OLS case. Let's first treat the case with $\lambda = 0$. Using $\widehat{Y} = X\widehat{\beta}$, $\widetilde{Y} = X\rho$, we compute:

$$\widehat{Y} = X(X^T X)^{-1} X^T \widetilde{Y} + \sqrt{1-\rho^2} X(X^T X)^{-1} X^T Z = \widetilde{Y} + \sqrt{1-\rho^2} X(X^T X)^{-1} X^T Z.$$

Note that $\mathbb{E}[\widetilde{Y}^2] = T\rho^2$. Then using the independence of X and Z:

$$\mathbb{E}[\widetilde{Y}^T \widehat{Y}] = \mathbb{E}[\rho^T X^T X \rho] = T\rho^2,$$

and:

$$\mathbb{E}[\widehat{Y}^2] = T\rho^2 + (1-\rho^2)\mathbb{E}[Z^T X(X^T X)^{-1} X^T Z] = T\rho^2 + (1-\rho^2)\mathbb{E}[Tr(X(X^T X)^{-1} X^T)].$$

Above the last expectation contains the trace of the $T \times T$ projection operator on the column space X, which is equal to $\min(N, T)$. Putting everything together we obtain:

$$\frac{\mathbb{E}[\widetilde{Y}^T \widehat{Y}]}{\sqrt{\mathbb{E}[\widetilde{Y}^2]\mathbb{E}[\widehat{Y}^2]}} = \frac{\rho}{\sqrt{\rho^2 + (1-\rho^2)\frac{\min(N,T)}{T}}}.$$

ii) Ridge case. Let's now move to the general case where $\lambda \geq 0$. For this we will use the singular value decomposition of $X = U\Sigma V^T$. Since X is a $T \times N$ real rectangular matrix, U, V are $T \times T$, $N \times N$ orthogonal matrices and Σ is a $T \times N$ diagonal matrix. The ridge estimator is given by the formula

$$\begin{aligned}\widehat{\beta} &= (X^T X + \lambda Id)^{-1} X^T Y \\ &= (X^T X + \lambda Id)^{-1} X^T \widetilde{Y} + \sqrt{1-\rho^2}(X^T X + \lambda Id)^{-1} X^T Z,\end{aligned}$$

which implies:

$$\widehat{Y} = X\widehat{\beta} = X(X^T X + \lambda Id)^{-1} X^T \widetilde{Y} + \sqrt{1-\rho^2} X(X^T X + \lambda Id)^{-1} X^T Z.$$

Recall that $\mathbb{E}[\widetilde{Y}^2] = T\rho^2$. We then need to compute the two quantities:

$$\mathbb{E}[\widetilde{Y}^T \widehat{Y}] = \mathbb{E}[\rho^T X^T X(X^T X + \lambda Id)^{-1} X^T X \rho] = \rho^T \mathbb{E}[V\Sigma^2(\Sigma^2 + \lambda Id)^{-1}\Sigma^2 V^T]\rho$$

$$\begin{aligned}\mathbb{E}[\widehat{Y}^2] &= \mathbb{E}[\rho^T X^T X(X^T X + \lambda Id)^{-1} X^T X(X^T X + \lambda Id)^{-1} X^T X \rho] \\ &+ (1-\rho^2)\mathbb{E}[Z^T X(X^T X + \lambda Id)^{-1} X^T X(X^T X + \lambda Id)^{-1} X^T Z] \\ &= \mathbb{E}[\rho^T V\Sigma^2(\Sigma^2 + \lambda Id)^{-1}\Sigma^2(\Sigma^2 + \lambda Id)^{-1}\Sigma^2 V^T \rho] \\ &+ (1-\rho^2)\mathbb{E}[Z^T U\Sigma(\Sigma^2 + \lambda Id)^{-1}\Sigma^2(\Sigma^2 + \lambda Id)^{-1}\Sigma U^T Z]\end{aligned}$$

To compute the expressions above, record the fact that for U an orthogonal matrix of size p distributed according to the Haar measure and D an independent diagonal matrix, one has:

$$\mathbb{E}[UDU^T] = \frac{1}{p}\mathbb{E}[tr(D)]Id_p.$$

One can then apply this formula to U, V as the orthogonal matrix and $f(\Sigma)$ for some f as the diagonal matrix D. The quantity $\mathbb{E}[tr(f(\Sigma))]$ can then be evaluated in the Marchenko-Pastur limit, namely the limit $N, T \to \infty$ with the ratio $N/T \to \gamma$, $\gamma \in (0, 1)$. It should then be possible to compute the terms above using the following integrals against the Marchenko-Pastur density f_{MP}:

$$\int_{x_-}^{x_+} f_{MP}(x)\frac{x^2}{x+\lambda}dx, \quad \int_{x_-}^{x_+} f_{MP}(x)\frac{x^2}{(x+\lambda)^2}dx, \quad \int_{x_-}^{x_+} f_{MP}(x)\frac{x^3}{(x+\lambda)^2}dx,$$

where $x_\pm = (1 \pm \sqrt{\gamma})^2$ and $f(x) = \frac{\sqrt{(x_+ - x)(x - x_-)}}{2\pi x\gamma}$. To be completed.

References

[1] George E. P. Box, Gwilym M. Jenkins, Gregory C. Reinsel, and Greta M. Ljung. *Time Series Analysis: Forecasting and Control.* John Wiley & Sons, 5 edition, 2015.

[2] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

[3] Abhimanyu Das, Weihao Kong, Andrew Leach, Shaan Mathur, Rajat Sen, and Rose Yu. Long-term forecasting with TiDE: Time-series Dense Encoder. In *arXiv preprint arXiv:2304.08424*, 2023.

[4] Jerome H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[5] Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. Axial attention in multidimensional transformers. arXiv preprint arXiv:1912.12180, 2019.

[6] Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting. In *International Conference on Learning Representations (ICLR)*, 2024. Spotlight.

[7] Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *International Conference on Learning Representations*, 2023.

[8] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

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

[10] Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. Transformers in time series: A survey. In Edith Elkind, editor, *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, pages 6878–6886. International Joint Conferences on Artificial Intelligence Organization, 2023. Survey Track.

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

[12] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 11121–11128, 2023.

[13] 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.

[14] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International Conference on Machine Learning*, pages 27268–27286. PMLR, 2022.