# Statistical benchmarking of transformer models in low signal-to-noise time-series forecasting

Cyril Garcia Guillaume Remy

February 10, 2026

## Abstract

We study the performance of transformer architectures for multivariate time-series forecasting in low-data regimes consisting of only a few years of daily observations. Using synthetically generated processes with known temporal and cross-sectional dependency structures and varying signal-to-noise ratios, we conduct bootstrapped experiments that enable direct evaluation via out-of-sample correlations with the optimal ground-truth predictor. We show that two-way attention transformers, which alternate between temporal and cross-sectional self-attention, can outperform standard baselines—Lasso, boosting methods, and fully connected multilayer perceptrons—across a wide range of settings, including low signal-to-noise regimes. We further introduce a dynamic sparsification procedure for attention matrices applied during training, and demonstrate that it becomes significantly effective in noisy environments, where the correlation between the target variable and the optimal predictor is on the order of a few percent. Analysis of the learned attention patterns reveals interpretable structure and suggests connections to sparsity-inducing regularization in classical regression, providing insight into why these models generalize effectively under noise.

## 1 Introduction

Time-series forecasting is the cornerstone of data-driven decision-making in multiple domains, from finance and meteorology to supply chain and energy management. Traditional statistical models, such as ARIMA and exponential smoothing Box et al. (2015), 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 Vaswani et al. (2017)—have shown promise in time-series tasks and have led to numerous architecture proposals for forecasting, as reviewed in Wen et al. (2023). Notable examples include the Informer model for efficient long-sequence forecasting Zhou et al. (2021), Autoformer with decomposition and auto-correlation mechanisms Wu et al. (2021), FEDformer leveraging frequency-enhanced attention Zhou et al. (2022), and PatchTST for patch-based representations Nie et al. (2023). 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 Zeng et al. (2023); Das et al. (2023) 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-section, or non-linear feature interactions), and *Order 2* (combined shifts, non-linear interactions in the time-series or cross-sectional dimension). 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 pre-processed and normalized. This allows us to focus solely on evaluating the transformers' ability to capture dependencies at different noise levels, leaving the pre-processing stage for a later study.

Our contributions are twofold. First, we test several transformer architectures with two-way attention mechanisms—two-way referring to applying attention along both the time-series and cross-sectional dimensions (as in Ho et al. (2019); Liu et al. (2024))—to exploit the multi-dimensional structure of time-series data. We benchmark these models against baselines like Lasso regression Tibshirani (1996), boosting Friedman (2001), as well as simpler neural network architectures, revealing that transformers outperform traditional methods on a number of dependencies and are general enough to work in several instances of data generating processes, even at high noise levels. Second, we propose and test a sparse attention implementation Child et al. (2019) to enhance robustness in low-signal regimes, which yields up to 10–20% performance gains in correlation with the optimal predictor. Additionally, we provide in Appendix B 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, model evaluation, and transformer architectures. Section 3 presents the experimental results when varying noise levels and time-series dependencies. Section 4 introduces our dynamic sparse attention implementation and presents the associated experimental results. Lastly, Section 5 gives a conclusion and future outlooks.

# 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 $X$:

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

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 $\epsilon$, 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 to have 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.*

## 2.2 Types of time-series dependencies

To make progress in understanding this forecasting problem we now make some assumptions on the function $\mathcal{F}$. We first rewrite (1) as $Y_{t,n} = \widetilde{Y}_{t,n} + \epsilon$, where $\widetilde{Y}_{t,n}$ is the ground-truth or optimal prediction of the model, and assume a correlation of $\rho = \text{Correl}(Y_{t,n}, \widetilde{Y}_{t,n})$. This $\rho$ is the key parameter that controls the signal-to-noise ratio in the experiments that we will run. We then further assume $\widetilde{Y}_{t,n}$ can be written as a sum of simple effects indexed by $e$:

$$\widetilde{Y}_{t,n} = \sum_e \rho_e \widetilde{Y}^e_{t,n}. \tag{2}$$

Here $\rho_e \in [0,1], \rho^2 = \sum_e \rho_e^2$, and $e$ will belong to a list of typical effects that are expected in a time-series forecasting problem which we now describe. The simplest effect $e = \text{Lin}$, which we will call the *Order 0*, corresponds to assuming the linear dependence

$$\widetilde{Y}^{\text{Lin}}_{t,n} = \sum_j \rho_{j,n} X^{(j)}_{t,n}, \tag{3}$$

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 linear case we can consider shifted relationships, either in the time-series (TS) or cross-section (CS) directions, namely

$$\widetilde{Y}^{\text{TS-Shift}}_{t,n} = \sum_j \rho_{j,n} X^{(j)}_{t-s_{j,n},n}, \tag{4}$$

$$\widetilde{Y}^{\text{CS-Shift}}_{t,n} = \sum_j \rho_{j,n} X^{(j)}_{t,n+k_{j,n}}, \tag{5}$$

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 (3) by a function $G : \mathbb{R}^F \to \mathbb{R}$ of the features, for $\rho_{\text{Fea-Nonlin}} \in [0,1]$:

$$\widetilde{Y}^{\text{Fea-Nonlin}}_{t,n} = \rho_{\text{Fea-Nonlin}}\, G(X^{(1)}_{t,n}, \ldots, X^{(F)}_{t,n}). \tag{6}$$

As a very basic example of $G$ we could take $X^{(j_1)}_{t,n}\text{sign}(X^{(j_2)}_{t,n})$, 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. $G(X^{(j_1)}_{t,n})$. We call (4), (5), and (6) the *Order 1* effects. Lastly we consider the following three effects

$$\widetilde{Y}^{\text{TSCS-Shift}}_{t,n} = \sum_j \rho_{j,n} X^{(j)}_{t-s_{j,n},n+k_{j,n}}, \tag{7}$$

$$\widetilde{Y}^{\text{TS-Nonlin}}_{t,n} = \sum_j \rho_{j,n} G(X^{(j)}_{t,n}, \ldots, X^{(j)}_{t-s,n}), \tag{8}$$

$$\widetilde{Y}^{\text{CS-Nonlin}}_{t,n} = \sum_j \rho_{j,n} G(X^{(j)}_{t,1}, \ldots, X^{(j)}_{t,N}), \tag{9}$$

which correspond respectively to shifting simultaneously in the time-series and cross-sectional directions, considering a non-linear interaction $G$ between time-series lags up to lag $t - s$, or a non-linear interaction $G$ in the cross-sectional direction. We classify these three effects as *Order 2*. It can be argued that since through equation (2) we are taking an arbitrary sum of all the effects described above, a very general class of dependencies $\mathcal{F}$ can be approximated.

## 2.3 Generating artificial data and model evaluation

Based on the hypotheses made above on $X, Y$ it is straightforward to generate this data with the following steps.

- Sample $X^{(j)}_{t,n}$ i.i.d. $\mathcal{N}(0,1)$ for all indices $t, n, j$.
- Following (2), combine the $X^{(j)}_{t,n}$ using one or a sum of effects presented above and obtain $\widetilde{Y}_{t,n}$.
- Assuming $\widetilde{Y}_{t,n}$ has been constructed to have mean 0 and variance $\rho^2 < 1$, define $Y_{t,n} = \widetilde{Y}_{t,n} + \sqrt{1-\rho^2} Z_{t,n}$, with $Z_{t,n}$ i.i.d. $\mathcal{N}(0,1)$.

This procedure gives an optimal prediction $\widetilde{Y}_{t,n}$ of mean 0 and variance $\rho^2$, and a target $Y_{t,n}$ of mean 0, variance 1, and correlation $\rho$ with $\widetilde{Y}_{t,n}$. A model will then, given $X^{(j)}_{s,n}$ for all $n, j$ and $s \leq t$, output $\widehat{Y}_{t,n}$, a prediction for $Y_{t,n}$. We split the data points into train and test, $T = T_{\text{train}} + T_{\text{test}}$, train the model using the $T_{\text{train}}$ points, and evaluate the model by computing the correlation between $\widehat{Y}_{t,n}$ and the optimal prediction $\widetilde{Y}_{t,n}$ using the $T_{\text{test}}$ points.

To fit our models we introduce a maximum lookback window length $T_{\text{win}}$, namely to make a prediction at time step $t$ the model will only use the $T_{\text{win}}$ time steps $t - T_{\text{win}} + 1 \leq s \leq t$. We also fix the dimensions of the data. Unless specified otherwise we use $T_{\text{train}} = 2500$, $T_{\text{test}} = 1500$, $N = 10$, $F = 20$, $T_{\text{win}} = 10$. Note that these dimensions are chosen such that a dataset containing daily observations over roughly ten years would be the same order of magnitude as the data studied here which should cover a rather wide range of practical examples. This is also constraining as this is a relatively small amount of data when compared to the usual experimental datasets studied (see Wu et al. (2021); Zhou et al. (2021) for references to many such datasets).

### 2.4 OLS, Lasso, Boosting, MLP as forecasting baseline

Before describing the different transformer architectures, we give a list of simple linear and non-linear models we can fit to obtain a simple performance baseline. In these benchmark models, since there is no explicit way to consider the time or cross-sectional dimensions, we fit those benchmarks by flattening the three dimensions $T_{\text{win}}, N, F$. This greedy approach considers a relatively large number of features and can be expected to be limited by the curse of dimensionality as dimensions increase, but this gives nevertheless a good baseline.

**i) Ordinary least squares (OLS).** First we can fit an OLS model ignoring the time-series structure. In this case, assuming only linear effects are present, one can theoretically compute the expected model performance. Let $\gamma$ denote the total number of features divided by the number of time points we have for prediction (fitting here one OLS model per time series to predict):

$$\gamma = \frac{T_{\text{win}} \cdot N \cdot F}{T_{\text{train}}}. \tag{10}$$

In Appendix B it is derived that the expected out-of-sample correlation to $\widetilde{Y}_{t,n}$ for this OLS prediction is given by:

$$C(\rho, \gamma) = \frac{\rho}{\sqrt{\rho^2 + (1-\rho^2)\frac{\gamma}{1-\gamma}}}. \tag{11}$$

We call this quantity TheoC in the result tables, it gives the expected amount of correlation when fitting an OLS per time-series using all the features.

**ii) Global Lasso regression / Boosting.** Next, we can run a global Lasso regression where the Lasso penalty parameter is estimated using a standard cross-validation. Here global refers to the fact that we treat all time-series as examples (instead of fitting one model per target time-series as in **i)**). More precisely, we use a unique set of regression betas $\beta_{s,n',j}$ indexed by lags, series, features to make all $N$ time-series prediction. As a simple non-linear ML benchmark we will also fit a standard boosting model using again all features, similarly as for the global Lasso. This gives a standard benchmark which has the possibility of capturing non-linear effects.

**iii) Global multi-layer perceptron.** Lastly, we move to a multi-layer perceptron (MLP), assuming no time-series structure, which we call the global MLP. This model flattens the input $X^{(j)}_{s,n} \in \mathbb{R}^{T_{\text{win}} \times N \times F}$ into a vector of dimension $T_{\text{win}} \times N \times F$ 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 taking advantage of the time-series structure.

### 2.5 Transformer architectures for forecasting

We now propose a general transformer architecture for time-series forecasting, 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 an input $\mathbf{X} \in \mathbb{R}^{B \times T_{\text{win}} \times N \times F}$ of dimensions batch size, lookback window length, number of time-series, number of features, the architecture proceeds as follows:

1. The feature dimension $F$ is linearly projected to $d_{\text{model}} = 64$.
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 $T_{\text{win}}$ dimension for each of the $N$ series (shape: $B \cdot N \times T_{\text{win}} \times d_{\text{model}}$).
- **C-block**: Independent cross-sectional attention over the $N$ variables at each time step (shape: $B \cdot T_{\text{win}} \times N \times d_{\text{model}}$).

Each attention uses 8 attention heads, and feeds into a feed-forward block with internal dimension 256.

4. An output head (LayerNorm $\to$ GELU $\to$ Linear) produces the final prediction for all $N$ targets.

A dropout of 0.1 is used for training. In the results table we call this model Trans, and unless stated otherwise it is run in the configuration `"TCTC"`, namely stacking twice a T-block followed by a C-block.

# 3 Experiments

We now present the results of the experiments. In all the tables below $\rho$ always corresponds to the global signal-to-noise ratio, namely the correlation between $\widetilde{Y}_{t,n}$ and $Y_{t,n}$. The rest of this section is split into two subsections, based on either fitting the models on a $Y_{t,n}$ generated from a single effect or on all the effects simultaneously.

## 3.1 Model performance on each individual effect

Here we construct our data $X, Y$ by choosing: $T_{\text{train}} = 2500$, $T_{\text{test}} = 1500$, $T_{\text{win}} = 10$, $N = 10$, $F = 20$. We pick as values of $\rho$: $2\%, 5\%, 10\%, 20\%, 50\%$. When building $\widetilde{Y}_{t,n}$ we only use a single effect, namely the sum (2) is restricted to a single effect $e$ in the list Lin, TS-Shift, CS-Shift, Fea-Nonlin, TSCS-Shift. Furthermore we only use half of the features when building $\widetilde{Y}_{t,n}$ from $X$, leaving the other half as noise. We give one result table for each effect and each entry in the tables corresponds to the out-of-sample performance of a model, which is measured as the correlation between the model prediction $\widehat{Y}_{t,n}$ and the ground-truth optimal prediction $\widetilde{Y}_{t,n}$, computed over the $T_{\text{test}}$ data points. Each result table contains five models as rows: TheoC corresponding to the value given by (11), Lasso, Boosting and MLP corresponding to the models presented in Section 2.4, and lastly our transformer model Trans described in Section 2.5. We now comment effect by effect on the results obtained:

- **Linear effect.** Generated from equation (3). Table 1 shows that Lasso regression is the best model in this case which is expected since we are dealing with a linear relationship, but the transformer model performs quite well at small $\rho$, especially compared to the Boosting and MLP.

- **TS-Shift effect.** Generated from equation (4) with the lag $s_{j,n} = 1$ for simplicity. Table 2 shows Lasso performs well here as in the linear case for the same reason, but the transformer model Trans does not perform quite as well on TS-Shift as on the Linear effect.

- **CS-Shift effect.** Generated from equation (5) where the cross-sectional shift $k_{j,n}$ is chosen at random for each $j, n$, namely each feature of each time-series predicts another time-series chosen among all possible. Table 3 shows that the transformer model has a great performance here, the cross-sectional attention detects this shift very well comparatively to the benchmarks.

- **Fea-Nonlin effect.** Generated from equation (6) with $G = X_{t,n}^{(j_1)} \text{sign}(X_{t,n}^{(j_2)})$, namely the conditioning of one feature by the sign of another. This is a non-linear effect. Table 4 shows that Lasso does not work here (expected), while the transformer performs very well. Both Boosting and MLP fail here because they flatten all features and the dimensionality is then too big to find the non-linear effect.

- **TSCS-Shift effect.** Generated from equation (7) using the same rules as above for both the TS and CS shifts, namely set $s_{j,n} = 1$ and choose $k_{j,n}$ at random. Table 5 shows that here interestingly the MLP is the best model, since it assumes no time-series structure it obtains similar results on the TS, CS, and TSCS shifts.

| $\rho$ | 0.02 | 0.05 | 0.10 | 0.20 | 0.50 |
|---|---|---|---|---|---|
| TheoC | 0.010 | 0.025 | 0.050 | 0.102 | 0.277 |
| Lasso | 0.038 | 0.305 | 0.785 | 0.940 | 0.995 |
| Boosting | 0.011 | 0.057 | 0.173 | 0.521 | 0.892 |
| MLP | 0.022 | 0.047 | 0.104 | 0.162 | 0.435 |
| Trans | 0.045 | 0.141 | 0.212 | 0.336 | 0.693 |

Table 1: Model performance for effect Linear

| $\rho$ | 0.02 | 0.05 | 0.10 | 0.20 | 0.50 |
|---|---|---|---|---|---|
| TheoC | 0.010 | 0.025 | 0.050 | 0.102 | 0.277 |
| Lasso | 0.038 | 0.315 | 0.812 | 0.960 | 0.996 |
| Boosting | 0.023 | 0.069 | 0.227 | 0.576 | 0.893 |
| MLP | 0.008 | 0.038 | 0.093 | 0.183 | 0.438 |
| Trans | 0.009 | 0.087 | 0.181 | 0.331 | 0.734 |

Table 2: Model performance for effect TS-Shift

| $\rho$ | 0.02 | 0.05 | 0.10 | 0.20 | 0.50 |
|---|---|---|---|---|---|
| TheoC | 0.010 | 0.025 | 0.050 | 0.102 | 0.277 |
| Lasso | 0.006 | 0.047 | 0.099 | 0.288 | 0.435 |
| Boosting | -0.002 | 0.020 | 0.023 | 0.061 | 0.214 |
| MLP | 0.020 | 0.044 | 0.089 | 0.189 | 0.419 |
| Trans | 0.018 | 0.050 | 0.118 | 0.218 | 0.589 |

Table 3: Model performance for effect CS-Shift

| $\rho$ | 0.02 | 0.05 | 0.10 | 0.20 | 0.50 |
|---|---|---|---|---|---|
| TheoC | 0.010 | 0.025 | 0.050 | 0.102 | 0.277 |
| Lasso | -0.001 | -0.010 | 0.001 | 0.005 | 0.001 |
| Boosting | 0.008 | -0.001 | -0.003 | -0.001 | 0.082 |
| MLP | 0.004 | -0.005 | -0.006 | -0.018 | -0.007 |
| Trans | 0.032 | 0.051 | 0.124 | 0.235 | 0.493 |

Table 4: Model performance for effect Fea-Nonlin

On the other hand the transformer architecture performs very poorly here.

Overall the transformer model strikes a good balance between capturing decently well linear effects and the TS-Shift, while being the only model detecting the non-linear feature interactions and having on average the best performance on the CS-shift. The only shortcoming is that it was not able to detect the TSCS-Shift effect. We leave understanding this problem for a future work.

### 3.2 Model performance on sum of all effects

Next we test the performance of our models at predicting a target $Y_{t,n}$ which simultaneously contains a superposition of 5 effects. Fix again $T_{\text{train}} = 2500$, $T_{\text{test}} = 1500$, $T_{\text{win}} = 10$, $N = 10$, $F = 20$. Now out of the 20 features, as before half of them will not be used to generate $\widetilde{Y}_{t,n}$, and out of the remaining 10 we attribute 2 to each of the five effects Lin, TS-Shift, CS- Shift, Fea-Nonlin, TSCS-Shift. Hence our $\widetilde{Y}_{t,n}$ contains the superposition of the 5 effects, but with a correlation to $Y_{t,n}$ still fixed to $\rho$. Now when testing the model performance, on top of giving as before the correlation between the model prediction $\widehat{Y}_{t,n}$ and $\widetilde{Y}_{t,n}$ ("Optimal" column in the result tables), we can also take a look at the correlation between $\widehat{Y}_{t,n}$ and each $\widetilde{Y}^e_{t,n}$ for each effect $e$ in the list of effects (column with the corresponding effect name in the result tables). This allows us to analyze which effect a model focuses on when all are present at once. For these results we set $\rho$ equal to 0.05 or 0.2 and obtain Table 6 and Table 7. These two tables show that the transformer model while not having the best correlation to the optimal prediction - Lasso is the best since it detects Linear and TS-Shift so well - shows a good performance simultaneously across all the effects at once, apart from the TSCS-shift it struggles to find.

## 4 Dynamic attention sparsity

### 4.1 Motivation

The experiments presented in the previous section suggest several intuitive properties that an optimal attention matrix should satisfy, depending on the structure of the underlying data-generating process. To make these intuitions explicit and simplify exposition, we focus on the transformer architecture composed of one temporal attention layer and one cross-sectional attention layer, i.e. a `"TC"` model in the notations of Section 2.5. We further restrict attention to a single head per layer, which allows direct visualization of the learned attention matrices.

| $\rho$ | 0.02 | 0.05 | 0.10 | 0.20 | 0.50 |
|---|---|---|---|---|---|
| TheoC | 0.010 | 0.025 | 0.050 | 0.102 | 0.277 |
| Lasso | 0.013 | 0.012 | 0.092 | 0.188 | 0.460 |
| Boosting | 0.008 | 0.005 | 0.014 | 0.049 | 0.221 |
| MLP | 0.023 | 0.037 | 0.089 | 0.182 | 0.433 |
| Trans | 0.009 | -0.008 | 0.001 | 0.005 | 0.029 |

Table 5: Model performance for effect TSCS-Shift

Importantly, the sparsity mechanisms discussed below naturally extend to deeper architectures and multi-head attention, although we do not explore these extensions here.

Several simple cases illustrate why sparsity may arise as a natural inductive bias for attention:

- **Temporal lag structure.** In the presence of a single predictive temporal lag (e.g., $s_{j,n} = 1$ in (5)), the optimal temporal attention matrix is sparse, with non-zero mass concentrated on the first sub-diagonal.
- **Multiple temporal lags.** If two lags carry predictive information (e.g., $s_{j,n} = 1$ or 2), the optimal attention matrix exhibits sparsity concentrated on the first two sub-diagonals.
- **Cross-sectional dependence.** When a single time series among many has predictive power for the target, the optimal cross-sectional attention matrix contains exactly one non-zero entry per row, corresponding to that series. The identity of this series is determined by the data-generating process and must be learned from the data.
- **Mixed effects.** In more general settings combining temporal and cross-sectional dependencies, attention matrices need not be sparse. In such cases, enforcing sparsity deterministically would be suboptimal.

These examples highlight a central challenge: while sparsity is often desirable, its structure cannot be specified without a prior and may vary substantially across regimes. This motivates the need for a mechanism that can adaptively sparsify attention when warranted by the data, while retaining dense attention when necessary.

### 4.2 Max attention algorithm

To improve robustness in low signal-to-noise regimes, we introduce a dynamic sparsification mechanism within the scaled dot-product attention computation. Unlike deterministic sparse attention patterns introduced in prior work (e.g., Child et al. (2019)), our approach is fully data-driven and does not assume a known sparsity structure. Moreover, it allows the effective sparsity level to vary across attention rows and across learning regimes.

The key idea is to treat each attention row as a discrete probability distribution and to assess the significance of its entries relative to the dominant mass in that row, rather than enforcing a fixed number of non-zero coefficients.

Given query, key, and value tensors $Q$, $K$, and $V$, we compute the attention logits

$$A = \frac{QK^\top}{\sqrt{d}} + b,$$

where $d$ denotes the head dimension and $b$ is a bias tensor encoding any attention mask (e.g., a causal mask with $b_{i,j} = -\infty$ for $j > i$).

We apply a softmax to compute attention probabilities and average these probabilities across the batch dimension, yielding an average attention matrix $\bar{P}$. For each row of $\bar{P}$, corresponding to a fixed head and query position, we compute its maximum entry $M$. Entries smaller than $K \cdot M$, where $0 < K < 1$ is a threshold parameter, are deemed insignificant and removed. This produces a binary mask $m$.

Using this mask, we construct an auxiliary bias tensor $b'$, initialized to zero and set to $-\infty$ wherever $m = 0$. The masked logits are then given by

$$A' = A + b',$$

with broadcasting over the batch dimension. The final attention weights are obtained by applying a softmax to $A'$, and are subsequently used to compute the weighted sum of values $V$.

In all experiments, we set $K = 0.1$. This parameter could also be learned during training or selected via cross-validation. We do not perform this optimization here.

Several alternative sparsification strategies—such as retaining the top-$k$ entries per row or thresholding based on quantiles—were considered. We

| Effect | Optimal | Linear | Fea-Nonlin | TS-Shift | CS-Shift | TSCS-Shift |
|---|---|---|---|---|---|---|
| Lasso | 0.103 | 0.178 | -0.009 | 0.067 | -0.005 | 0.011 |
| Boosting | 0.024 | 0.013 | -0.000 | 0.036 | 0.002 | 0.004 |
| MLP | 0.035 | 0.017 | 0.002 | 0.018 | 0.028 | 0.013 |
| Trans | 0.065 | 0.069 | 0.022 | 0.029 | 0.031 | -0.004 |

Table 6: Model performance on superposition of effects, $\rho = 0.05$

| Effect | Optimal | Linear | Fea-Nonlin | TS-Shift | CS-Shift | TSCS-Shift |
|---|---|---|---|---|---|---|
| Lasso | 0.605 | 0.590 | 0.021 | 0.594 | 0.084 | 0.085 |
| Boosting | 0.257 | 0.251 | 0.020 | 0.251 | 0.033 | 0.029 |
| MLP | 0.139 | 0.078 | 0.013 | 0.072 | 0.069 | 0.084 |
| Trans | 0.225 | 0.160 | 0.118 | 0.113 | 0.102 | 0.018 |

Table 7: Model performance on superposition of effects, $\rho = 0.2$

found these approaches less suitable for two reasons. First, they impose a fixed sparsity level, implicitly assuming prior knowledge of the true dependency structure. Second, under causal masking, the effective length of attention rows varies with the query position, leading such methods to remain dense near the top of the attention matrix while becoming sparse closer to the bottom.

Our design is motivated by simple limiting cases. In a pure lag-1 TS-shift, the dominant sub-diagonal entry converges toward probability one, while all others remain close to zero. Introducing a second predictive lag leads the two corresponding entries to converge toward equal mass (e.g., 0.5 each). These cases illustrate the need for a relative, rather than absolute, sparsification criterion.

Overall, this procedure adaptively filters weak or noisy attention links while preserving high-confidence dependencies, without imposing a fixed sparsity pattern.

### 4.3 Bootstrap results

To isolate the effect of dynamic sparsification, we first consider the simplest setting of a lag-1 TS-Shift. As discussed above, the optimal temporal attention matrix in this case contains a single non-zero sub-diagonal. We compare three otherwise identical transformer configurations:

- A full attention **full_attention** transformer with no enforced sparsity.
- A deterministically sparse **deterministic_sparse** transformer in which the temporal attention matrix is fixed to its optimal structure. Since this requires knowledge of the data-generating process, it serves as an upper bound on achievable performance.
- A transformer equipped with the dynamic sparsification algorithm **max_sparse** described above.

To assess statistical significance, we perform a two-sample hypothesis test comparing the distribution of out-of-sample correlations obtained by the full-attention model and by the dynamically sparse model. Each sample corresponds to a single iteration of a bootstrap in which the dataset is fully regenerated with identical statistical properties. The significance of the means of the populations is assessed using the p-value of the test **p_value**. We repeat this procedure across a range of signal-to-noise ratios in order to quantify how the benefits of sparsity depend on the difficulty of the forecasting problem. For each bootstrapped performance metric we consider a number of samples $n = 90$.

We start by considering the case of the TS-Shift effect with lag 1 for which we have an upper bound benchmark of the model with a deterministic sparsity on the sub-diagonal in Table 8.

| $\rho$ | 0.015 | 0.03 | 0.1 |
|---|---|---|---|
| full_attention | 0.017 | 0.071 | 0.455 |
| max_sparse | 0.025 | 0.083 | 0.456 |
| deterministic_sparse | 0.096 | 0.200 | 0.469 |
| p_value | 0.028 | 0.126 | 0.467 |

Table 8: Sparsity performance for effect TS-Shift

| $\rho$ | 0.015 | 0.03 | 0.1 |
|---|---|---|---|
| full_attention | 0.012 | 0.034 | 0.101 |
| max_sparse | 0.019 | 0.038 | 0.102 |
| p_value | 0.021 | 0.094 | 0.463 |

Table 9: Sparsity performance for effect CS-Shift

| $\rho$ | 0.015 | 0.03 | 0.1 |
|---|---|---|---|
| full_attention | 0.035 | 0.075 | 0.243 |
| max_sparse | 0.034 | 0.075 | 0.231 |
| p_value | 0.638 | 0.464 | 0.953 |

Table 10: Sparsity performance on superposition of effects

| $\rho$ | 0.015 | 0.03 | 0.1 |
|---|---|---|---|
| full_attention | 0.135 | 0.255 | 0.538 |
| max_sparse | 0.120 | 0.245 | 0.530 |
| p_value | 0.902 | 0.779 | 0.724 |

Table 11: Sparsity performance for Linear effect

As expected, a deterministic sparsity informed by the ground truth generating process gives the best results. It can be noted that the p-value computed to consider the difference of performance between **max_sparse** and **full_attention** shows high level of significance for lower correlations and monotonically increases with $\rho$.

We can then consider a similar setup, the CS-Shift effect for which we don't have a readily informed deterministic sparse structure given the ground truth model in Table 9. We see very similar results with a 10% significance for both $\rho = 0.015$ and $\rho = 0.03$ with the same monotonic increase in p-value.

Now considering both the superposition of effects and the simple linear effects, we don't expect much added value from a learned sparisty but one can hope that the model remains competitive with a performance that can be compared with a dense attention matrix. This is indeed what we observe as highlighted in Table 10 and Table 11 which shows the robustness and the ability to learn the sparsity structure of a given prediction problem.

As a way to visualize what happens in the temporal and cross-sectional cases, we present in Appendix A the attention matrix learned within each generating process for one random example of the bootstrap samples. The temporal attention matrix for a TS-Shift effect is shown in Table 12 and the cross-sectional attention matrix for a CS-Shift effect is shown in Table 13. Note that while the temporal attention shape is fairly intuitive, the cross-sectional attention would have only one non-zero coefficient per row in the absence of noise. We see here how the sparsity helps to remove a part of the noise for both examples (often one clear dominating coefficient per row in the CS case and the sub-diagonal of the temporal matrix with high probability weight).

## 5 Conclusion and outlooks

In this work, we adopted a statistical approach based on synthetically generated canonical effects to study both the performance and the underlying mechanisms of transformer architectures for multivariate time-series forecasting. Our objective was to help bridge the gap between the rapidly growing empirical literature on transformers—often centered on fixed benchmark real-world datasets—and more interpretable statistical analysis that facilitates mechanistic understanding and generalization to complex settings. Our results show that, even in regimes characterized by extremely low signal-to-noise ratios, transformer architectures with stacked temporal and cross-sectional attention layers can outperform traditional time-series models. At the same time, we observe that interactions between these two dimensions — corresponding in practice to shifts simultaneously in the time-series and cross-sectional directions — remain more challenging to capture, even when multiple attention layers are

combined. This limitation points to promising directions for future research on how different attention dimensions should be composed or integrated. Motivated by an analogy with the curse of dimensionality in classical statistics, we introduced a dynamic sparsification mechanism that can be applied independently to each attention layer and head. Empirically, this mechanism yields significant improvements over dense attention in settings where sparsity is optimal, while remaining competitive when the data-generating process does not exhibit sparse structure. These benefits are particularly pronounced in very low signal-to-noise regimes. Beyond the specific architectural insights, our study highlights the value of synthetic data and controlled experimental designs for assessing statistical significance and inductive biases in modern deep learning models, especially in settings with limited data. We believe this perspective may inform both real-world time-series forecasting applications and future methodological work aimed at understanding and improving transformer-based models.

GitHub repository available at:
`https://github.com/cyrilgarcia009/TSNN`

## References


Box, G. E. P., Jenkins, G. M., Reinsel, G. C., and Ljung, G. M. *Time Series Analysis: Forecasting and Control*. John Wiley & Sons, 5th edition, 2015.

Child, R., Gray, S., Radford, A., and Sutskever, I. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Das, A., Kong, W., Leach, A., Mathur, S., Sen, R., and Yu, R. Long-term forecasting with TiDE: Time-series Dense Encoder. *arXiv preprint arXiv:2304.08424*, 2023.

Friedman, J. H. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 29(5):1189–1232, 2001.

Ho, J., Kalchbrenner, N., Weissenborn, D., and Salimans, T. Axial attention in multidimensional transformers. *arXiv preprint arXiv:1912.12180*, 2019.

Liu, Y., Hu, T., Zhang, H., Wu, H., Wang, S., Ma, L., and Long, M. itransformer: Inverted transformers are effective for time series forecasting. In *International Conference on Learning Representations*, 2024.

Nie, Y., Nguyen, N. H., Sinthong, P., and Kalagnanam, J. A time series is worth 64 words: Long-term forecasting with transformers. In *International Conference on Learning Representations*, 2023.

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

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems*, volume 30, 2017.

Wen, Q., Zhou, T., Zhang, C., Chen, W., Ma, Z., Yan, J., and Sun, L. Transformers in time series: A survey. *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, pp. 6778–6786, 2023.

Wu, H., Xu, J., Wang, J., and Long, M. Autoformer: Decomposition transformers with autocorrelation for long-term series forecasting. In *Advances in Neural Information Processing Systems*, pp. 22419–22430, 2021.

Zeng, A., Chen, M., Zhang, L., and Xu, Q. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, pp. 11121–11128, 2023.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., and Zhang, W. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, pp. 11106–11115, 2021.

Zhou, T., Ma, Z., Wen, Q., Wang, X., Sun, L., and Jin, R. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International Conference on Machine Learning*, pp. 27268–27286. PMLR, 2022.

# A Examples of sparse attention matrices

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| 1 | 0.650 | 0.350 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| 2 | 0.141 | 0.616 | 0.243 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| 3 | 0.000 | 0.129 | 0.637 | 0.235 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| 4 | 0.000 | 0.000 | 0.000 | 0.751 | 0.249 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| 5 | 0.000 | 0.000 | 0.061 | 0.135 | 0.567 | 0.237 | 0.000 | 0.000 | 0.000 | 0.000 |
| 6 | 0.000 | 0.000 | 0.000 | 0.000 | 0.112 | 0.682 | 0.206 | 0.000 | 0.000 | 0.000 |
| 7 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.133 | 0.658 | 0.210 | 0.000 | 0.000 |
| 8 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.141 | 0.631 | 0.228 | 0.000 |
| 9 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.075 | 0.144 | 0.573 | 0.209 |

Table 12: Temporal learned attention matrix for $\rho = 0.03$

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.714 | 0.286 |
| 1 | 0.000 | 0.000 | 0.000 | 0.135 | 0.453 | 0.140 | 0.000 | 0.000 | 0.273 | 0.000 |
| 2 | 0.245 | 0.000 | 0.000 | 0.000 | 0.000 | 0.157 | 0.000 | 0.218 | 0.000 | 0.380 |
| 3 | 0.030 | 0.242 | 0.118 | 0.048 | 0.000 | 0.216 | 0.000 | 0.176 | 0.030 | 0.139 |
| 4 | 0.526 | 0.000 | 0.000 | 0.000 | 0.000 | 0.256 | 0.139 | 0.000 | 0.000 | 0.079 |
| 5 | 0.047 | 0.237 | 0.087 | 0.179 | 0.046 | 0.068 | 0.096 | 0.000 | 0.141 | 0.098 |
| 6 | 0.272 | 0.000 | 0.155 | 0.146 | 0.000 | 0.049 | 0.112 | 0.000 | 0.267 | 0.000 |
| 7 | 0.081 | 0.000 | 0.069 | 0.000 | 0.279 | 0.506 | 0.000 | 0.000 | 0.065 | 0.000 |
| 8 | 0.116 | 0.269 | 0.000 | 0.198 | 0.098 | 0.036 | 0.056 | 0.059 | 0.168 | 0.000 |
| 9 | 0.122 | 0.000 | 0.000 | 0.000 | 0.064 | 0.150 | 0.445 | 0.048 | 0.000 | 0.170 |

Table 13: Cross-sectional learned attention matrix for $\rho = 0.03$

# B Analytical derivation of the expected OLS performance

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, number of time-series lags used), we 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 $\widetilde{Y} := \sum_{i=1}^{N} \rho_i X_i$ and $\rho^2 := \sum_{i=1}^{N} \rho_i^2 < 1$. Define $Y = \widetilde{Y} + \sqrt{1-\rho^2} Z$ 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 $\beta_i$ using OLS as $\widehat{\beta} = (X^T X)^{-1} X^T Y$ and from here we obtain the model prediction

$\widehat{Y} = X\widehat{\beta}$. The goal is thus to understand the distribution of Correl$(\widehat{Y}, \widetilde{Y})$ as a function of $N, T$. Here we will just compute the expectation.

**i) In-sample correlation.** We assume that $N \leq T$ so that $\widehat{\beta}$ is almost surely well-defined. Using $\widehat{Y} = X\widehat{\beta}$, $\widetilde{Y} = X\rho$, we compute:

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

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

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

and:

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

The trace in the expectation above simplifies to the trace of the identity matrix of size $N$, which simply gives $N$. Putting everything together we obtain:

$$\frac{\mathbb{E}[\widetilde{Y}^T\widehat{Y}]}{\sqrt{\mathbb{E}[\widetilde{Y}^2]\mathbb{E}[\widehat{Y}^2]}} = \frac{\rho}{\sqrt{\rho^2 + (1-\rho^2)\frac{N}{T}}}. \tag{12}$$

**ii) Out-of-sample correlation.** We repeat the above computation but computing this time an out-of-sample correlation. More precisely, consider now $X_o$ a $T_o \times N$ matrix with $\mathcal{N}(0,1)$ i.i.d. entries independent of everything. Setting $\widehat{Y}_o = X_o\widehat{\beta}$, $\widetilde{Y}_o = X_o\rho$, the goal is now to compute Correl$(\widehat{Y}_o, \widetilde{Y}_o)$ as a function of $N, T, T_o$. In this case we get $\mathbb{E}[\widetilde{Y}_o^2] = T_o\rho^2$, $\mathbb{E}[\widetilde{Y}_o^T\widehat{Y}_o] = T_o\rho^2$, and:

$$\mathbb{E}[\widehat{Y}_o^2] = T_o\rho^2 + (1-\rho^2)\mathbb{E}[Z^TX(X^TX)^{-1}X_o^TX_o(X^TX)^{-1}X^TZ].$$

In the last term the expectation is over the randomness of $Z, X, X_o$ which are all independent. The expectation over $Z$ gives a trace as before and we arrive at:

$$\mathbb{E}[\widehat{Y}_o^2] = T_o\rho^2 + (1-\rho^2)\mathbb{E}[Tr(X_o^TX_o(X^TX)^{-1})] = T_o\rho^2 + T_o(1-\rho^2)\mathbb{E}[Tr((X^TX)^{-1})].$$

To evaluate the last expectation above, we use the Marchenko-Pastur limit, i.e. $N, T \to \infty$ with $N/T \to \gamma < 1$, under which $Tr((X^TX)^{-1})$ converges almost surely to $\frac{\gamma}{1-\gamma}$. Putting everything together we obtain under this limit:

$$\frac{\mathbb{E}[\widetilde{Y}_o^T\widehat{Y}_o]}{\sqrt{\mathbb{E}[\widetilde{Y}_o^2]\mathbb{E}[\widehat{Y}_o^2]}} \to \frac{\rho}{\sqrt{\rho^2 + (1-\rho^2)\frac{\gamma}{1-\gamma}}}. \tag{13}$$