WHAT MATTERS IN DEEP LEARNING FOR TIME SERIES FORECASTING?

Anonymous authors
Paper under double-blind review

ABSTRACT

Deep learning models have grown increasingly popular in time series applications. However, the large quantity of newly proposed architectures, together with often contradictory empirical results, makes it difficult to assess which components contribute significantly to final performance. We aim to make sense of the current design space of deep learning architectures for time series forecasting by discussing the design dimensions and trade-offs that can explain, often unexpected, observed results. We discuss the necessity of grounding model design on principles for forecasting groups of time series and how such principles can be applied to current models. In particular, we assess how concepts such as locality and globality apply to recent forecasting architectures. We show that accounting for these aspects can be more relevant for achieving accurate results than adopting specific sequence modeling layers and that simple, well-designed forecasting architectures can often match the state of the art. We discuss how overlooked implementation details in existing architectures (1) fundamentally change the class of the resulting forecasting method and (2) drastically affect the observed empirical results. Our results call for rethinking current faulty benchmarking practices and for the need to focus on the foundational aspects of the forecasting problem when designing neural network architectures. As a step in this direction, we also propose an *auxiliary forecasting model card*, i.e., a template with a set of fields to characterize existing and new forecasting architectures based on key design choices.

1 INTRODUCTION

Novel sequence modeling architectures are consistently improving the state-of-the-art in many applications (Gu et al., 2022; Gu and Dao, 2024; Beck et al., 2024), such as text and natural language processing. However, results in time series forecasting offer a much more uncertain way ahead, with recent work questioning the effectiveness of modern deep learning approaches (Toner and Darlow, 2024; Zeng et al., 2023; Tan et al., 2024). The result is that current research is seemingly stuck in a loop of positive results being quickly dismissed by new evidence that questions our understanding of the components that contribute to obtaining accurate forecasts (Shao et al., 2024). Recent works propose several architectures, e.g., based on attention (Zhou et al., 2021; Wu et al., 2021; Nie et al., 2023; Liu et al., 2023a; Zhang and Yan, 2023; Liu et al., 2022a), and assess their performance against state-of-the-art methods on common benchmarks. Most of these architectures are obtained by stacking and combining different components and operators and involve many–often hidden–implementation choices (e.g., parameter sharing and local parameters). However, the impact of such design choices on the resulting model and its performance is often overlooked. As an example, in recent works, a collection of synchronous time series is often considered as a single multivariate signal. This approach can lead to misconceptions and results that are difficult to interpret. Starting from this consideration, recent architectures stemming from Nie et al. (2023) rely on what has been called *channel-independence*, i.e., on processing each channel of a time series independently from others while sharing the same parameters. This approach has–somewhat surprisingly–led to superior results when compared to standard multivariate models. However, when these channels correspond to different univariate and homogeneous (related) time series (as is often the case in commonly used benchmarks), "channel-independence" corresponds to adopting the framework of global models, which is well understood in time series analysis (Benidis et al., 2022; Salinas et al., 2020; Januschowski et al., 2020; Montero-Manso and Hyndman, 2021). While this

might sound simply an issue of naming conventions, it can provide clear explanations for observed results (Montero-Manso and Hyndman, 2021) and unlock new designs (Wang et al., 2019; Smyl, 2020). For example, there is a large body of literature on methods that account for dependencies among synchronous time series while keeping (part of) the model global (Cini et al., 2023; Sen et al., 2019). As an additional example, other works have recently observed that applying attention among channels can improve performance (Liu et al., 2023a; Zhang and Yan, 2023). However, if we consider the different channels as a collection of correlated time series, similar attention operators have been routinely used in spatiotemporal forecasting models (Ma et al., 2019; Grigsby et al., 2021; Marisca et al., 2022; Liu et al., 2023b). Besides missed opportunities and insights, these aspects can also harm the effectiveness of our benchmarking practices. Indeed, overlooked design choices can, as we will show, lead to empirical results that are difficult to interpret and that might mislead the designer. Moreover, as we will discuss throughout the paper, those mentioned are only a selection of the issues that contribute to the current situation.

In this paper, we scan the design space of modern deep learning architectures for time series forecasting and assess the impact that associated design choices have on current benchmarking practices. In particular, we aim at understanding the state of the field by relying on well-understood principles for forecasting groups of time series. In doing so, we examine recent architectures and empirical results, highlighting the impact of overlooked aspects that are often considered as mere implementation details. To frame the discussion, we structure our analysis by considering four main dimensions: **D1.**) **model configuration**, i.e., selecting the model family (e.g., local, global, or hybrid); **D2.**) **preprocessing and exogenous variables**, i.e., selecting exogenous variables and setting up preprocessing and postprocessing operations; **D3.**) **temporal processing**, i.e., accounting for temporal (i.e., intra-series) dependencies. **D4.**) **spatial processing**, i.e., accounting for spatial (i.e., inter-series) dependencies. While some of these dimensions are not always orthogonal (e.g., space and time contributions can be processed in an integrated way), we believe that analyzing how these different aspects concur to characterize a model family is the key to understanding recent results. We argue that, to assess meaningful improvements to the state of the art, any comparison must ensure that design choices in any of these dimensions do not interfere with the evaluation of the proposed component. In this context, our contributions are as follows.

- We analyze the current state of deep learning for time series forecasting by relying on principles to forecast groups of time series to make sense of often contradictory empirical results.
- We empirically assess the impact of overlooked design choices and implementation details in existing state-of-the-art architectures, and show that they explain a significant portion of the observed performance improvements.
- We show that a streamlined architecture built on well-understood design principles can match the performance of the state-of-the-art.
- To move forward, we introduce an *auxiliary forecasting model card* template–complementary to existing generic model cards (Mitchell et al., 2019)–that can be used to characterize existing and new forecasting architectures.

The current trends in the field have led to focusing on finding a one-size-fits-all architecture with state-of-the-art performance in benchmarks. This prompted the adoption of increasingly more complex architectures that combine many poorly understood components. By showing the limitations of common benchmarking practices, our work is aimed at stimulating discussion on our current approach to conducting machine learning for time series forecasting. We believe that this discussion is an important step for the maturity of the field and to ensure future progress.

2 Related work and context

The history of neural networks in forecasting applications is long, and has often been characterized by skepticism (Zhang et al., 1998). However, the forecasting community is reaching consensus on the effectiveness of deep learning methods in settings where a single neural network can be trained on (large) collections of related time series (Hewamalage et al., 2021; Benidis et al., 2022). Models based on this approach have been called *global* in contrast with *local* models, which are instead trained separately on each time series (Montero-Manso and Hyndman, 2021; Januschowski et al., 2020; Benidis et al., 2022). Global models and hybrid global-local variants thereof have won

forecasting competitions (Smyl, 2020) and been adopted by the industry (Salinas et al., 2020; Kunz et al., 2023). With the increase in popularity of new sequence modeling architectures (Vaswani et al., 2017; Gu et al., 2022; Gu and Dao, 2024; Orvieto et al., 2023), the machine learning community has started investigating how to adapt such architectures to the forecasting problem. In particular, the Informer (Zhou et al., 2021) architecture is among the first of a line of works aiming at tailoring Transformers (Vaswani et al., 2017) to long-range time series forecasting. Together with the architecture, Zhou et al. (2021) also introduced a popular benchmark where collections of time series are considered as a single multivariate sequence. Several subsequent works follow, then, the same approach (Wu et al., 2021; Liu et al., 2022a; Wu et al., 2023; Liu et al., 2022b; Zhou et al., 2022). Zeng et al. (2023) and Toner and Darlow (2024) show that most of these architectures can be outperformed in such benchmarks by simple linear models. Nie et al. (2023), then, showed that–in the same settings–superior results could be achieved by processing each channel independently with shared parameters. For many of these benchmarks, this essentially corresponds to the global approach; indeed, a large part of the associated datasets consists of collections of related time series, even though, as already mentioned, they have often been seen as a single multivariate sequence. Follow-up works (Liu et al., 2023a; Zhang and Yan, 2023) then reintroduced components to model dependencies across multiple time series while keeping the core of the model global. Conflating the problem of forecasting any group of time series into forecasting a single multivariate sequence, as we will see, can be problematic and lead to unclear designs (Sec. 4.1). Moreover, current popular architectures stack several components and rely on many hidden implementation choices, which make a direct comparison of introduced sequence modeling operators challenging.

The need to clarify inconsistencies in benchmarking practices has pushed researchers to focus on developing new benchmarks and evaluation pipelines for forecasting (Shao et al., 2024; Wang et al., 2024a; Qiu et al., 2024). Conversely, in our work, we aim to assess whether the performance gains of the recently proposed method come from the use of specific operators or lie in other implementation details. Said differently, we aim to assess whether current benchmarking practices are focusing on *what really matters* in deep learning for time series forecasting. Similar analysis has been done in other subfields of machine learning, such as reinforcement learning (Raichuk et al., 2021), and in the context of graph neural networks (GNNs) (Errica et al., 2020; Dwivedi et al., 2023).

3 PRELIMINARIES

3.1 PROBLEM SETTING

We consider a collection of N time series $\mathcal{D} = \{\boldsymbol{x}^1_{0:L_1}, \dots, \boldsymbol{x}^N_{0:L_N}\}$, where $\boldsymbol{x}^i_{0:L_i} \in \mathbb{R}^{L_i \times d_x}$ denotes the sequence of L_i d_x-dimensional observations associated with the i-th time series. When present, exogenous variables are denoted as $\boldsymbol{u}^i_{0:L_i} \in \mathbb{R}^{L_i \times d_u}$. A binary mask, $\boldsymbol{m}^i_{0:L_i} \in \{0,1\}^{L_i \times d_x}$, may be introduced to model missing or invalid observations. Time series in the set can come from different domains and be generated by different stochastic processes. In such a setting, the mask $\boldsymbol{m}^i_{0:L_i}$ can be used to account for heterogeneous time series by modeling missing channels. If time series are synchronous, we use capital letters to denote values across the collection, e.g., $\boldsymbol{X}_t \in \mathbb{R}^{N \times d_x}$ refers to the stacked observations at time step t. Time series in the collection might be *correlated* (in a broad sense), i.e., uncertainty on future values of each time series might be reduced by taking into account observations from other time series.

Forecasting groups of time series We consider the problem of multi-step ahead time series forecasting, i.e., the problem of predicting the next $H \geq 1$ observations $\boldsymbol{x}^i_{t:t+H}$ for the i-th time series, given a window $W \geq 1$ of past observations $\boldsymbol{x}^i_{t-W:t}$ from the same time series. As the stochastic process generating data p^i is unknown, the objective is to approximate it with a model $p_{\boldsymbol{\theta}}$ with learnable parameters $\boldsymbol{\theta}$ such that

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}^i_{t:t+H} \mid \boldsymbol{x}^i_{t-W:t}, \boldsymbol{u}^i_{t-W:t}, \boldsymbol{u}^i_{t:t+H}) \approx p^i(\boldsymbol{x}^i_{t:t+H} \mid \boldsymbol{x}^i_{<t}, \boldsymbol{u}^i_{<t}, \boldsymbol{u}^i_{t:t+H}) \quad \forall\, i = 1, \dots, N \tag{1}$$

where $\boldsymbol{x}^i_{<t}$ denotes all past observations of the i-th series preceding timestep t. We focus on the problem of obtaining *point forecasts* $\widehat{\boldsymbol{x}}_{t:t+H}$ of, e.g., the expected value such as $\widehat{\boldsymbol{x}}^i_{t:t+H} \approx \mathbb{E}_p\left[\boldsymbol{x}^i_{t:t+H}\right]$ by using a parametric model $\mathcal{F}(\,\cdot\,;\boldsymbol{\theta})$. Predictions are obtained by fitting parameters $\boldsymbol{\theta}$ of the chosen model family. As we will discuss in Sec. 4.1, we say that a model is *global* if its parameters are shared across all the time series. In such a case, the model is trained on the entire set of

time series. Conversely, a model is *local* if its parameters are specific to a single time series. If relying on local models, forecasting a collection of time series results in fitting a separate model for each sequence in the set. Choosing between a local and global approach (or a hybrid thereof) depends on the task at hand, data availability, and model complexity. As mentioned in Sec. 1, due to advantages in sample efficiency, global models are a particularly appealing choice when relying on deep learning architectures (Hewamalage et al., 2021; Benidis et al., 2022). Additionally, global models can be employed inductively, e.g., in a cold start scenario, whereas local models are transductive. We will expand this discussion in Sec. 4.1.

3.2 Baselines

Through the paper, we assess the impact of different design choices with respect to a set of recent state-of-the-art architectures for long-range time series forecasting, which we compare against simpler, streamlined baselines (see Sec. 3.2). We consider representative models that have shaped the development of recent time series forecasting methods and that demonstrate competitive performance on benchmarks. We include: 1. **PatchTST** (Nie et al., 2023), the widely used architecture that–as already mentioned–introduced "channel independence" and patching-based Transformer layers. In particular, PatchTST relies on splitting the input into fixed-size patches before applying attention; 2. **DLinear** (Zeng et al., 2023), which combines a linear model with a time series decomposition step; 3. **TimeMixer** (Wang et al., 2024b), which is an multilayer perceptron (MLP)-based architecture processing the input at different resolutions; 4. **Linear**, a linear autoregressive model trained with $L2$ regularization and ordinary least squares (OLS), following Toner and Darlow (2024), implemented in both its global and local variants. We also consider models that incorporate spatial processing: 5. **iTransformer** (Liu et al., 2023a), which processes the temporal dynamics with a feedforward layer and then uses standard attention among channels; 6. **ModernTCN** (Donghao and Xue, 2024) which employs convolutional layers for spatio-temporal representation; 7. **Crossformer** (Zhang and Yan, 2023), which uses patching and spatiotemporal attention operators to model dependencies among different channels of the input time series. To ensure a fair comparison, we evaluate all the models in the same benchmarking setup, under unified settings, and with access to the same exogenous variables. We rely on the available open-source implementations of each approach and adapt them to our evaluation procedure and standardized inputs. The code for all the experiments, based on the Torch Spatiotemporal library (Cini and Marisca, 2022), will be open-sourced upon publication. For a more detailed description of each baseline, we refer the reader to App. A.

Reference architectures In our experiments, we compare state-of-the-art models against a reference streamlined architecture specifically designed to assess the impact of different design choices w.r.t. the target design dimensions. Note that the purpose here is not to propose a new architecture to challenge the state of the art. Conversely, reference architectures provide baselines, introduced to facilitate a fair and consistent comparison and to gauge the impact of different design choices more directly. For the temporal module, we consider several alternatives: a MLP with residual connections, a temporal convolutional network (TCN) with causal dilated filters (Bai et al., 2018), a gated recurrent neural network (RNN) (Chung et al., 2014), a stack of Transform layers (Vaswani et al., 2017), and pyramidal attention operators akin to the Pyraformer architecture (Liu et al., 2022a). In the tables, we denote these reference models as **MLP**, **TCN**, **RNN**, **Transf.**, **Pyraf.**, respectively. For the TCN, RNN and attention-based models, we use a 1-D convolutional layer with a large stride as an additional preprocessing step to implement an operator akin to patching (Nie et al., 2023) and facilitate the processing at the subsequent layers. For the spatial module, we use a simple spatial attention layer (denoted as **sp. attn.**). For additional implementation details, please refer to App. B.

3.3 Benchmarks

As a benchmark, we use four real-world datasets from different domains that are widely used in the context of long-range time series forecasting (Wang et al., 2024b; Zhang and Yan, 2023; Liu et al., 2023a; Zeng et al., 2023; Nie et al., 2023). In particular: **Electricity** collects hourly electricity usage for 321 customers (Wu et al., 2021); **Weather** includes 21 meteorological variables collected every 10 minutes from Germany (Wu et al., 2021); **Traffic** contains hourly road occupancy data collected from various 862 sensors on San Francisco's highways (Wu et al., 2021); **Solar** contains 10-minute records of solar power generation from 137 photovoltaic plants (Lai et al., 2018). We use a $70\%/10\%/20\%$

split for the training, validation, and testing, following previous works (Wang et al., 2024b). Metrics are reported on scaled data for consistency with published benchmarks. All the numerical results are averaged over three independent runs. We use a window size of 96 for all experiments, while in Tab. 3 we use a longer window size of 336 (except for Solar). For further details, refer to App. C.

4 What Matters in Deep Learning for Time Series Forecasting?

In this section, we go through four key design dimensions that characterize forecasting architectures and have a significant impact on overall performance. For each design dimension, we assess how different choices have contributed to making the current progress in the field uncertain, leading to often unexpected empirical results.

D1. Model configuration This refers to the type of forecasting model being employed. We distinguish between local models (each trained on individual time series), global models (trained on multiple series jointly with the same shared weights), and hybrid approaches that combine elements of both paradigms.

D2. Preprocessing and exogenous variables This dimension refers to the transformations applied to the data either before or after being used as input to a predictor, and to the exogenous variables used as additional inputs to the forecasting architecture.

D3. Temporal processing Temporal processing refers to the operators used to model temporal dependencies within an architecture.

D4. Spatial processing This dimension involves mechanisms used to model inter-series dependencies when multiple time series are available as inputs.

We do not aim to provide an exhaustive discussion of each dimension, but instead focus on how they have been addressed in recent research and highlight their impact on performance.

4.1 Design dimension 1: Model configuration

As previously discussed, the model configuration–global, local, or hybrid –is a fundamental aspect in model design, since it radically changes the type of model being used. Yet, this aspect is often left unspecified or dealt with as an implementation detail. However, choosing between a local, global, or hybrid approach has several implications that should be properly discussed (Montero-Manso and Hyndman, 2021; Januschowski et al., 2020; Salinas et al., 2020). For instance, it is often common to model any collection of synchronous time series as a single highly-dimensional multivariate time series and hence consider models such as

$$\widehat{\boldsymbol{X}}_{t:t+H} = \mathcal{F}\left(\boldsymbol{X}_{t-W:t}, \dots; \boldsymbol{\theta}\right). \tag{2}$$

However, this approach can scale poorly with the input's dimensionality. Indeed, in practice, several recent works (e.g., Nie et al. (2023); Liu et al. (2023a)) have observed that processing each channel independently with the same parameters empirically yields better performance. As already mentioned, this is equivalent to adopting the well-known global approach, i.e., to processing related time series as

$$\widehat{\boldsymbol{x}}^i_{t:t+H} = \mathcal{F}\left(\boldsymbol{x}^i_{t-W:t}, \dots; \boldsymbol{\theta}\right) \quad \forall\, i = 1, \dots, N. \tag{3}$$

Moreover–although not always explicitly stated–several architectures, e.g., (Wang et al., 2024b; Zhang and Yan, 2023; Donghao and Xue, 2024), adopt the approach in Eq. 3, but introduce some time series specific parameters $\boldsymbol{\phi}^i$, resulting in models

$$\widehat{\boldsymbol{x}}^i_{t:t+H} = \mathcal{F}\left(\boldsymbol{x}^i_{t-W:t}, \dots; \boldsymbol{\theta}, \boldsymbol{\phi}^i\right) \quad \forall\, i = 1, \dots, N. \tag{4}$$

which effectively consist of hybrid global-local models (Smyl, 2020; Cini et al., 2023; Benidis et al., 2022). The design choices that, in practice, lead to models as in Eq. 4 are often dealt with as implementation details. For instance, Wang et al. (2024b) uses learnable local parameters in the normalization module; Salinas et al. (2020)–while relying on an otherwise global model–uses a different one-hot-encoding vector associated with each processed time series, effectively introducing a vector of learnable parameters specific to that input sequence. Finally, an opposite trend seen in other approaches–often relying on simple (linear) models (Zeng et al., 2023)–design models in Eq. 2 by using different parameters for each time series

$$\widehat{\boldsymbol{x}}^i_{t:t+H} = \mathcal{F}\left(\boldsymbol{x}^i_{t-W:t}, \dots; \boldsymbol{\theta}^i\right) \quad \forall\, i = 1, \dots, N, \tag{5}$$

hence yielding to *local models*.

Clearly, models in Eq. 2–5 correspond to fundamentally different approaches that can result in markedly different performance. Failing to recognize the impact of the associated design choices can be problematic for several reasons. First, the use of shared versus local parameters may have very different effects depending on whether the time series are homogeneous (e.g., data coming from identical sensors at different locations) or heterogeneous (e.g., measurements of different physical quantities). Moreover, when dealing with multiple multivariate time series, a multivariate global model is often more appropriate than a univariate one that processes channels independently. Second, as we will see, comparing the results of models belonging to different typologies without stating it explicitly can make it difficult to interpret performance differences. In Tab. 1, we assess the performance–in terms of mean squared error (MSE)–of different architectures from the literature on a set of benchmarks (see Sec. 3.3). As in all our experiments, we focus on the task of long-range time series forecasting, with a prediction horizon of 96 time steps. We evaluate changes in performance for the reference Transformer and for two architectures that incorporate local parameters by removing these components, and, conversely, for the iTransformer by adding them. As one would expect, using local parameters drastically changes the observed results. Mixing results from the two columns without accounting for the impact of these design choices would clearly lead to misleading conclusions. Therefore, when aiming to identify the most effective sequence modeling operators, experiments should be designed to factor out the impact of model configuration.

Table 1: Comparison (MSE) of models with local embeddings. Best average results are in **bold**.

D	Model	Hybrid	Global
Electr.	Transf.	**0.136**$_{\pm.000}$	0.151$_{\pm.000}$
	Crossformer	**0.141**$_{\pm.001}$	0.146$_{\pm.003}$
	TimeMixer	**0.151**$_{\pm.000}$	0.180$_{\pm.001}$
	iTransformer	**0.139**$_{\pm.000}$	0.154$_{\pm.000}$
Weather	Transf.	**0.153**$_{\pm.001}$	0.177$_{\pm.002}$
	Crossformer	**0.154**$_{\pm.003}$	0.164$_{\pm.003}$
	TimeMixer	**0.164**$_{\pm.002}$	0.178$_{\pm.001}$
	iTransformer	**0.154**$_{\pm.000}$	0.170$_{\pm.001}$
Traffic	Transf.	0.417$_{\pm.009}$	**0.392**$_{\pm.000}$
	Crossformer	0.540$_{\pm.014}$	**0.512**$_{\pm.007}$
	TimeMixer	0.464$_{\pm.001}$	**0.463**$_{\pm.001}$
	iTransformer	0.435$_{\pm.002}$	**0.409**$_{\pm.000}$
Solar	Transf.	**0.196**$_{\pm.000}$	0.205$_{\pm.001}$
	Crossformer	0.177$_{\pm.008}$	**0.166**$_{\pm.005}$
	TimeMixer	**0.366**$_{\pm.017}$	0.367$_{\pm.017}$
	iTransformer	**0.189**$_{\pm.001}$	0.197$_{\pm.002}$

4.2 Design dimension 2: Preprocessing and exogenous variables

Exogenous variables and preprocessing (e.g., scaling, detrending, and methods accounting for seasonality) are ingredients that can have a significant impact on final performance. In this section, we discuss how exogenous variables and preprocessing methods have been considered and included inconsistently across popular baselines and benchmarks. Similar to model configuration, these benchmarking practices further prevent a clear understanding of the reasons behind the observed performances. This issue is compounded by the growing trend of comparing newly proposed architectures directly against published results of existing methods, without reproducing those results in this exercise. It follows that differences in preprocessing routines become increasingly difficult to isolate and account for. To investigate the extent of this problem in recent benchmarks, we focus specifically on the use of exogenous variables. For instance, PatchTST, DLinear, and Crossformer do not use covariates by default, while iTransformer does. We then evaluate the impact of adding the same covariates (calendar features, in this case) to some of these baselines and report the outcome of this experiment in Tab. 2. Results show the impact of including covariates on the performance of models such as DLinear, PatchTST and Crossformer, which do exclude them in their original implementations, as well as on iTransformer and the reference Transformer, which include them. Their effect is more pronounced on some

Table 2: Comparison (MSE) of models with and without covariates. Best average results are in **bold**

D	Model	w/ exog.	w/out exog.
Electr.	Transf.	**0.136**$_{\pm.000}$	0.155$_{\pm.001}$
	PatchTST	**0.128**$_{\pm.000}$	0.134$_{\pm.000}$
	Crossformer	**0.139**$_{\pm.002}$	0.141$_{\pm.001}$
	iTransformer	**0.154**$_{\pm.000}$	0.167$_{\pm.000}$
	DLinear	**0.193**$_{\pm.000}$	0.195$_{\pm.000}$
Weather	Transf.	**0.153**$_{\pm.001}$	0.161$_{\pm.000}$
	PatchTST	**0.174**$_{\pm.000}$	0.180$_{\pm.002}$
	Crossformer	0.154$_{\pm.003}$	0.154$_{\pm.003}$
	iTransformer	**0.170**$_{\pm.001}$	0.176$_{\pm.000}$
	DLinear	0.199$_{\pm.005}$	**0.196**$_{\pm.001}$
Traffic	Transf.	**0.417**$_{\pm.009}$	0.479$_{\pm.006}$
	PatchTST	**0.355**$_{\pm.000}$	0.383$_{\pm.001}$
	Crossformer	0.548$_{\pm.024}$	**0.540**$_{\pm.014}$
	iTransformer	**0.409**$_{\pm.000}$	0.444$_{\pm.001}$
	DLinear	**0.609**$_{\pm.000}$	0.648$_{\pm.000}$
Solar	Transf.	**0.196**$_{\pm.000}$	0.206$_{\pm.003}$
	PatchTST	**0.196**$_{\pm.001}$	0.225$_{\pm.003}$
	Crossformer	**0.176**$_{\pm.006}$	0.177$_{\pm.008}$
	iTransformer	**0.197**$_{\pm.002}$	0.221$_{\pm.003}$
	DLinear	**0.246**$_{\pm.001}$	0.285$_{\pm.001}$

Table 3: Forecasting results (MSE and MAE) for a horizon of 96 steps for models *not including* spatial processing. Best average results are in **bold**, second best are underlined.

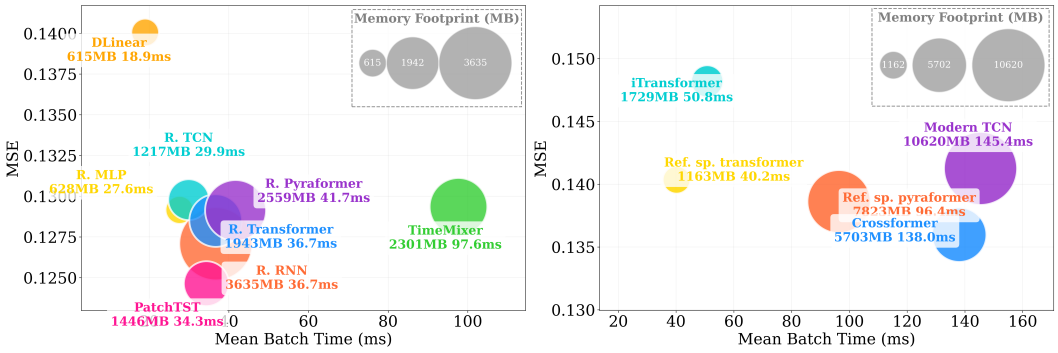
MODEL	Electricity		Weather		Traffic		Solar	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Linear Global	0.140	0.237	0.174	0.234	0.410	0.282	0.222	0.291
Linear Local	0.134	0.230	**0.144**	0.209	0.426	0.298	0.223	0.295
MLP	0.129±.000	0.225±.000	0.148±.001	0.198±.000	0.376±.000	0.253±.001	0.194±.003	0.239±.002
RNN	0.147±.001	0.247±.001	0.149±.001	0.203±.001	0.390±.007	0.275±.002	0.200±.003	0.246±.004
TCN	0.130±.000	0.224±.000	0.148±.000	0.200±.001	0.364±.003	0.253±.002	0.193±.004	0.243±.005
Transf.	0.129±.001	0.222±.001	0.149±.001	0.203±.002	0.362±.003	0.249±.002	0.203±.006	0.245±.002
Pyraf.	0.129±.001	0.224±.001	0.148±.001	0.199±.001	0.365±.002	0.251±.003	**0.189±.003**	**0.236±.004**
TimeMixer	0.129±.001	0.224±.000	0.147±.001	0.197±.000	0.373±.002	0.271±.003	0.199±.001	0.245±.000
PatchTST	**0.125±.000**	**0.218±.000**	0.148±.001	**0.195±.001**	**0.345±.000**	**0.234±.000**	0.197±.001	0.244±.004
DLinear	0.140±.000	0.237±.000	0.173±.000	0.232±.001	0.407±.000	0.283±.000	0.246±.001	0.331±.000

benchmarks while less evident in others. These results pinpoint another source of uncertainty in interpreting recent benchmarking results; preprocessing steps should be standardized across baselines to ensure that all models have access to the same inputs.

4.3 DESIGN DIMENSION 3: TEMPORAL PROCESSING

In this section, we assess whether streamlined models, properly configured as hybrid global-local models with exogenous inputs and local embeddings, can achieve performance comparable to that of recent state-of-the-art models. This design dimension, concerning sequence modeling operators, has been the main focus of recent research. However, this line of work has produced contrasting results, leading to considerable confusion about which components effectively contribute to performance (Toner and Darlow, 2024; Zeng et al., 2023; Tan et al., 2024). We focus on methods that only process inputs along the temporal dimension, while approaches that include components accounting for spatial dependencies are discussed in Sec. 4.4. For this analysis, we use the different reference architectures introduced in Sec. 3.2, and compare them against three popular and well-established baselines, namely DLinear, PatchTST, and TimeMixer, by using standardized inputs (including covariates), and hyperparameter tuning. We remark that the goal is not to determine which architecture performs best, but rather to assess the extent to which different temporal processing components influence the observed results. As shown in Tab. 3, no single model consistently outperforms the others. Moreover, reference architectures that rely on standard and simple operators obtain competitive performance against the state of the art across all the considered scenarios. These results challenge the effectiveness of current benchmarking practices in identifying the components responsible for performance improvements and in measuring the contribution brought by the different sequence modeling operators. Additionally, results show that, in many scenarios, choosing a specific sequence modeling operator is not the critical design choice. Analogous observations are confirmed in Sec. 4.4.

Note that in all experiments, we process data from the Weather dataset as if it were a collection of univariate time series, to show the effect of handling it as is commonly done in the literature. Interestingly, one of the best-performing models on Weather is the local OLS linear model. This is not too surprising, since Weather is actually a multivariate time series with heterogeneous channels, and among the models in Tab. 3, that linear model is the only one that explicitly models each time series as heterogeneous. Although results do not provide a clear ordering of methods, we did observe that patching, which, as mentioned earlier, corresponds to applying a nonlinear convolutional filter over time, works well across both reference architectures and state-of-the-art baselines, providing a good approach for enabling the processing of long input windows. Hierarchical attention-based approaches (such as Pyraformer) also showed to be a viable option. Finally, we report in Fig. 1 an assessment of the computational scalability of the different architectures, in terms of time needed to process each batch and GPU memory usage. The computational cost is visualized in relation to the forecasting accuracy. Results show that MLP, TCN, and PatchTST achieve a good trade-off between MSE performance, GPU memory usage, and training time. We encourage conducting analysis like this to gain insight into the most suitable models for given benchmarks.



(a) Models *not including* spatial processing. (b) Models *including* spatial processing.

Figure 1: MSE versus mean batch time during training on the Electricity dataset. Circle size indicates memory consumption.

Table 4: Forecasting results (MSE) for a horizon of 96 steps for models *including* spatial processing. Best average results are in **bold**, second best are underlined.

(a) Comparison of reference architectures with spatial attention against state-of-the-art baselines.

Model	Electricity	Weather	Traffic	Solar
MLP + sp. attn.	0.140±.001	0.157±.000	0.435±.006	0.201±.009
Pyraf. + sp. attn.	0.139±.001	0.157±.002	**0.389±.002**	0.188±.002
iTransformer	0.148±.000	0.171±.001	0.393±.001	0.208±.003
Crosformer	**0.136±.000**	**0.152±.003**	0.527±.002	**0.184±.008**
ModernTCN	0.141±.000	0.154±.001	0.445±.001	0.190±.001

(b) **Ablation:** iTransformer with or without spatial attention.

	iTransformer	
Dataset	Spatial att.	Feedforward
Electricity	**0.148±.000**	0.149±.001
Weather	**0.171±.001**	**0.171±.000**
Traffic	0.393±.001	**0.390±.001**
Solar	0.208±.003	**0.194±.001**

4.4 Design dimension 4: Spatial processing

We call *spatial* the dimension that spans multiple time series, which may correspond to different spatial locations when considering physical sensors. This section complements the discussion started in Sec. 4.3 by considering models that account for inter-series dependencies by relying on different operators. In particular, we compare the reference architectures, where dependencies are modeled with a standard spatial Transformer, against three state-of-the-art baselines: iTransformer, Crossformer, and ModernTCN. For the reference architectures, we use an MLP or pyramidal attention for temporal processing. Tab. 4a reports the results of the comparison where we reduced the length of the input window associated with each time series to keep computational costs manageable. Analogously to Sec. 4.3, simulations show that the simple, streamlined architectures perform comparably to the state of the art, highlighting once again the limitations of current benchmarking practices. The results in Tab. 3 and Tab. 4a, combined with the observation that spatial dependencies might provide limited benefits in long-range forecasting, have led us to doubt the effectiveness of spatial attention operators in this context. For this reason, we design an additional architecture by removing the spatial attention layer, replaced by a simple MLP, in iTransformer, effectively removing all the components modeling spatial dependencies in the architecture. Results in Tab. 4b indeed show that in this context, entirely removing spatial attention led to better or similar performance in all the considered datasets. These observations further highlight the need for more thorough assessments of how each component contributes to the observed results. Finally, Fig. 1 reports performance in relation to computational cost, which in this case is particularly critical as processing data along the spatial dimension can have a significant impact on computational scalability.

5 Discussion

The results in Sec. 4.1–4.4 question whether we have been successful in measuring the advances of deep learning architectures for time series forecasting and shed light on several faults in current benchmarking practices. We showed that overlooked design choices can have a significant impact and that

simple, well-designed architectures can match the state of the art on standard benchmarks (see Tab. 3 and 4a). Our analysis calls for reaching a better understanding of the architecture's design space, showing how misconceptions in model specification can trigger misleading conclusions (as shown, e.g., in Tab. 1). The additional ablation studies (e.g., in Tab. 2 and 4b) corroborate these findings that are further substantiated by plenty of additional empirical results in the appendix of the paper. However, our objective is not to be dismissive of the progress of the field–which is tangible in many applications–but rather to ensure that we can move forward by focusing on answering foundational questions and fostering awareness of the existing flaws in our practices. Revisiting the benchmarking pipeline will be a crucial step in this direction. For instance, introducing benchmarks specifically designed to isolate distinct dimensions, possibly by relying on synthetic datasets, can help measure the isolated effects of different design choices. Moreover, model cards (Mitchell et al., 2019) can be an effective tool, providing a simple and practical way to summarize a model's main characteristics and to facilitate model comparison. Below, we propose a set of *forecasting model cards* that can be used in conjunction with existing model cards to capture relevant aspects of the design dimensions discussed in the paper, and we provide an example of its use in App. D. We believe that by recalibrating our evaluation tools on reliably measuring actual progress, the shift toward answering more foundational methodological questions would happen as a natural consequence.

FORECASTING MODEL CARD

Model setting

- Size of the input window
- Whether the model is transductive or inductive, and can be used in a cold start scenario
- How to mask missing observations and/or if imputation is needed

D1. Model configuration

- Whether the model is global, local, or hybrid
- *If the model is hybrid*, which parameters are shared across the time series and which are not

D2. Preprocessing and exogenous variables

- The type of scaling or other transformation applied at training and inference time
- Temporal covariates, lagged variables, or other types of exogenous variables are employed

D3. Temporal processing

- Modules and operators used to encode observations along the temporal axis
- Time and space complexity w.r.t. the length of the time series being processed

D4. Spatial processing

If spatial dependencies are accounted for:

- Modules used to model spatial dynamics and whether a graph structure is employed
- Time and space complexity w.r.t. the number of the time series being processed

6 CONCLUSION

We investigated the effectiveness of current benchmarks in measuring progress in the field and the impact of design choices on forecasting performance. We showed, by analyzing points of failure in our evaluation procedures, that our current practices might produce misleading results. With this paper, we pinpointed several of the sources of this uncertainty and aimed to foster a discussion to ensure that the field can move forward and address its current limitations. Indeed, our analysis also shows that appropriate design choices do have an impact on performance and can explain seemingly contradictory empirical results. We believe that the results and the analysis presented in this paper are an important step toward focusing on *what matters* in deep learning for time series forecasting.

Limitations In this work, we focused on a specific set of dimensions in the design space of forecasting models that have had a strong impact on the benchmarking results of recent studies.

Clearly, this analysis can be extended to other aspects of the design space. For example, the discussion can be expanded to include probabilistic forecasting and the choice of metrics to quantify forecasting accuracy. Moreover, it could cover additional aspects included in the forecasting model card template, such as missing values, graph structures, and transductive versus inductive settings. Additionally, future work could explore similar issues in short-term forecasting benchmarks.

REFERENCES

Albert Gu, Karan Goel, and Christopher Re. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=uYLFoz1vlAC.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024.

Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xlstm: Extended long short-term memory. *Advances in Neural Information Processing Systems*, 37:107547–107603, 2024.

William Toner and Luke Nicholas Darlow. An analysis of linear time series forecasting models. In *International Conference on Machine Learning*, pages 48404–48427. PMLR, 2024.

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.

Mingtian Tan, Mike A Merrill, Vinayak Gupta, Tim Althoff, and Thomas Hartvigsen. Are language models actually useful for time series forecasting? In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=DV15UbHCY1.

Zezhi Shao, Fei Wang, Yongjun Xu, Wei Wei, Chengqing Yu, Zhao Zhang, Di Yao, Tao Sun, Guangyin Jin, Xin Cao, et al. Exploring progress in multivariate time series forecasting: Comprehensive benchmarking and heterogeneity analysis. *IEEE Transactions on Knowledge and Data Engineering*, 2024.

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.

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*, 2021.

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.

Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting. *arXiv preprint arXiv:2310.06625*, 2023a.

Yunhao Zhang and Junchi Yan. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=vSVLM2j9eie.

Shizhan Liu, Hang Yu, Cong Liao, Jianguo Li, Weiyao Lin, Alex X. Liu, and Schahram Dustdar. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *International Conference on Learning Representations*, 2022a. URL https://openreview.net/forum?id=0EXmFzUn5I.

Konstantinos Benidis, Syama Sundar Rangapuram, Valentin Flunkert, Yuyang Wang, Danielle Maddix, Caner Turkmen, Jan Gasthaus, Michael Bohlke-Schneider, David Salinas, Lorenzo Stella, François-Xavier Aubet, Laurent Callot, and Tim Januschowski. Deep learning for time series forecasting: Tutorial and literature survey. *ACM Comput. Surv.*, 55(6), dec 2022. ISSN 0360-0300. doi: 10.1145/3533382. URL https://doi.org/10.1145/3533382.

David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3): 1181–1191, 2020.

Tim Januschowski, Jan Gasthaus, Yuyang Wang, David Salinas, Valentin Flunkert, Michael Bohlke-Schneider, and Laurent Callot. Criteria for classifying forecasting methods. *International Journal of Forecasting*, 36(1):167–177, 2020.

Pablo Montero-Manso and Rob J Hyndman. Principles and algorithms for forecasting groups of time series: Locality and globality. *International Journal of Forecasting*, 37(4):1632–1653, 2021.

Yuyang Wang, Alex Smola, Danielle Maddix, Jan Gasthaus, Dean Foster, and Tim Januschowski. Deep factors for forecasting. In *International conference on machine learning*, pages 6607–6617. PMLR, 2019.

Slawek Smyl. A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International journal of forecasting*, 36(1):75–85, 2020.

Andrea Cini, Ivan Marisca, Daniele Zambon, and Cesare Alippi. Taming local effects in graph-based spatiotemporal forecasting. *Advances in Neural Information Processing Systems*, 36:55375–55393, 2023.

Rajat Sen, Hsiang-Fu Yu, and Inderjit S Dhillon. Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting. *Advances in neural information processing systems*, 32, 2019.

Jiawei Ma, Zheng Shou, Alireza Zareian, Hassan Mansour, Anthony Vetro, and Shih-Fu Chang. Cdsa: cross-dimensional self-attention for multivariate, geo-tagged time series imputation. *arXiv preprint arXiv:1905.09904*, 2019.

Jake Grigsby, Zhe Wang, Nam Nguyen, and Yanjun Qi. Long-range transformers for dynamic spatiotemporal forecasting. *arXiv preprint arXiv:2109.12218*, 2021.

Ivan Marisca, Andrea Cini, and Cesare Alippi. Learning to Reconstruct Missing Data from Spatiotemporal Graphs with Sparse Observations. In *Advances in Neural Information Processing Systems*, 2022.

Hangchen Liu, Zheng Dong, Renhe Jiang, Jiewen Deng, Jinliang Deng, Quanjun Chen, and Xuan Song. Spatio-temporal adaptive embedding makes vanilla transformer sota for traffic forecasting. In *Proceedings of the 32nd ACM international conference on information and knowledge management*, pages 4125–4129, 2023b.

Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. Model cards for model reporting. In *Proceedings of the conference on fairness, accountability, and transparency*, pages 220–229, 2019.

Guoqiang Zhang, B Eddy Patuwo, and Michael Y Hu. Forecasting with artificial neural networks:: The state of the art. *International journal of forecasting*, 14(1):35–62, 1998.

Hansika Hewamalage, Christoph Bergmeir, and Kasun Bandara. Recurrent neural networks for time series forecasting: Current status and future directions. *International Journal of Forecasting*, 37 (1):388–427, 2021.

Manuel Kunz, Stefan Birr, Mones Raslan, Lei Ma, and Tim Januschowski. Deep learning based forecasting: a case study from the online fashion industry. In *Forecasting with artificial intelligence: theory and applications*, pages 279–311. Springer, 2023.

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

Antonio Orvieto, Samuel L Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. Resurrecting recurrent neural networks for long sequences. In *International Conference on Machine Learning*, pages 26670–26698. PMLR, 2023.

Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. In *International Conference on Learning Representations*, 2023.

Yong Liu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Non-stationary transformers: Exploring the stationarity in time series forecasting. *Advances in neural information processing systems*, 35: 9881–9893, 2022b.

Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *Proc. 39th International Conference on Machine Learning (ICML 2022)*, 2022.

Yuxuan Wang, Haixu Wu, Jiaxiang Dong, Yong Liu, Mingsheng Long, and Jianmin Wang. Deep time series models: A comprehensive survey and benchmark. 2024a.

Xiangfei Qiu, Jilin Hu, Lekui Zhou, Xingjian Wu, Junyang Du, Buang Zhang, Chenjuan Guo, Aoying Zhou, Christian S Jensen, Zhenli Sheng, et al. Tfb: Towards comprehensive and fair benchmarking of time series forecasting methods. *CoRR*, 2024.

Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphaël Marinier, L'eonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, and Sylvain Gelly. What matters for on-policy deep actor-critic methods? a large-scale study. In *International Conference on Learning Representations*, 2021. URL https://api.semanticscholar.org/CorpusID:233340556.

Federico Errica, Marco Podda, Davide Bacciu, and Alessio Micheli. A fair comparison of graph neural networks for graph classification. In *Proceedings of the 8th International Conference on Learning Representations (ICLR)*, 2020.

Vijay Prakash Dwivedi, Chaitanya K Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *Journal of Machine Learning Research*, 24 (43):1–48, 2023.

Shiyu Wang, Haixu Wu, Xiaoming Shi, Tengge Hu, Huakun Luo, Lintao Ma, James Y Zhang, and JUN ZHOU. Timemixer: Decomposable multiscale mixing for time series forecasting. In *International Conference on Learning Representations (ICLR)*, 2024b.

Luo Donghao and Wang Xue. ModernTCN: A modern pure convolution structure for general time series analysis. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=vpJMJerXHU.

Andrea Cini and Ivan Marisca. Torch Spatiotemporal, 3 2022. URL https://github.com/TorchSpatiotemporal/tsl.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pages 95–104, 2018.

Taesung Kim, Jinhee Kim, Yunwon Tae, Cheonbok Park, Jang-Ho Choi, and Jaegul Choo. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=cGDAkQo1C0p.

Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009. ISBN 1441412697.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

William Falcon and The PyTorch Lightning team. PyTorch Lightning, 3 2019. URL `https://github.com/PyTorchLightning/pytorch-lightning`.

Omry Yadan. Hydra - a framework for elegantly configuring complex applications. Github, 2019. URL `https://github.com/facebookresearch/hydra`.

Charles R Harris, K Jarrod Millman, Stéfan J Van Der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al. Array programming with numpy. *Nature*, 585(7825):357–362, 2020.

Matthias Fey, Jinu Sunil, Akihiro Nitta, Rishi Puri, Manan Shah, Blaž Stojanovič, Ramona Bendias, Barghi Alexandria, Vid Kocijan, Zecheng Zhang, Xinwei He, Jan E. Lenssen, and Jure Leskovec. PyG 2.0: Scalable learning on real world graphs. In *Temporal Graph Learning Workshop @ KDD*, 2025.

APPENDIX

A BASELINES

Table 5: Description for the baseline models

Model	Model configuration	Temporal processing	Spatial processing
Dlinear	Global	Linear layers	Not modeled
PatchTST	Global	Temporal convolution followed by temporal attention over the patches	Not modeled
TimeMixer	Hybrid	Feedforward networks applied to the trend and seasonal components, downsampled at different scales	Not modeled
Crossformer	Hybrid	Temporal convolution followed by attention applied over the patches, with a hierarchical structure constructed with linear layers	Spatial attention applied among patches of different time series
iTransformer	Global	Feedforward layers	Spatial attention applied among different time series
ModernTCN	Hybrid	Depth-wise convolutions	Convolution applied across time series
Linear global/local	Global/local	Linear autoregression	Not modeled

Below, we provide a brief description of each baseline as employed in our experiments on the considered benchmarks. Furthermore, we summarize them in Tab. 5 using three fields corresponding

to the design dimensions introduced in Sec. 4, excluding the *preprocessing and exogenous variables* dimension, since the considerable differences among the methods make it less informative.

- Dlinear (Zeng et al., 2023) decomposes the input into seasonal and trend components using a moving average and processes them with linear layers. The hyperparameters determine its local-global nature. In the table, we report it as global because, in our experiments, it was used in this configuration. We follow the same convention for PatchTST and TimeMixer.
- PatchTST (Nie et al., 2023) has strongly influenced subsequent works by employing a global Transformer, in contrast to earlier local multivariate approaches that treated the group of input time series as a single multivariate series. PatchTST segments the time series and generates corresponding embeddings using an operation analogous to temporal convolution. Then, it applies attention over these segments, referred to as *patches*. It does not model spatial relations.
- TimeMixer (Wang et al., 2024b) is a fully MLP-based architecture that downsamples the input at different scales, decomposes it into trend and seasonal components, and employs feedforward layers to model temporal dependencies.
- Crossformer (Zhang and Yan, 2023) employs an input encoding with segmentation analogous to that used in PatchTST. The model is a hybrid global-local model, as it includes learnable position embeddings for each time series in the set. In addition to temporal attention, it captures spatial dependencies through attention over the spatial dimension using a routing mechanism. Furthermore, it adopts a hierarchical encoder-decoder structure.
- iTransformer (Liu et al., 2023a) is a global model that uses a feedforward approach to encode temporal dynamics and spatial attention to model spatial dependencies. This method has been described as applying attention to the *inverted dimension*, i.e., the spatial dimension. The model is global.
- ModernTCN (Donghao and Xue, 2024) uses depth-wise convolutions to encode temporal information, with an encoding similar to that performed in PatchTST, and then applies point-wise convolutions to process the feature and spatial dimensions separately.
- Linear (Toner and Darlow, 2024) is linear autoregressive models trained with $L2$ regularization and OLS. The *local* variant employs different weights for each series, while the *global* variant employs the same weights for all the series in the set.

B Reference architectures structure

The reference architectures, as schematized in Fig. 2, consist of a preprocessing module, followed by the processing of the temporal and spatial dynamics, and finally a postprocessing module. Their modular structure facilitates understanding of the architecture and promotes fair comparisons, as each module can be modified independently. In our experiments, we kept most modules fixed, modifying only the temporal and spatial modules for the experiments reported in Tables 3 and 11, respectively, and occasionally the feature encoding module. Here, we provide a more detailed description than the one given in Sec. 3.2.

Preprocessing module The preprocessing module begins with RevInv (Kim et al., 2022) normalization. Then, the feature encoding moduleprocesses the input and covariates through non-linear layers and returns their sum. Alternatively, it can perform temporal convolution to generate an encoding similar to that in (Nie et al., 2023). Finally, local embeddings are concatenated with the resulting encoding.

Processing module The processing module consists of temporal processing followed by spatial processing. In a more general architecture, these components could be interleaved. For simplicity, however, they are treated separately in our implementation.

Postprocessing module The postprocessing module consists of a linear decoder that maps the hidden representations to predictions for the horizon. Finally, the predictions are de-normalized using the RevInv module.

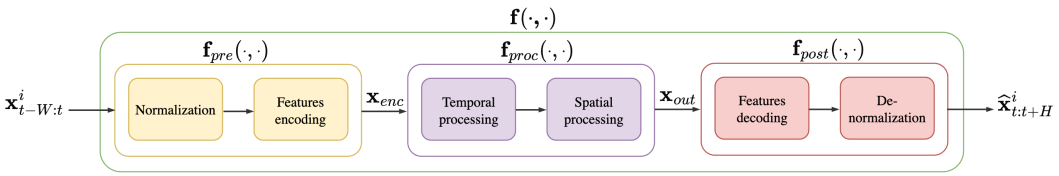


Figure 2: Block diagram of the reference architectures

C Empirical setup and additional experimental results

Our code is implemented in Python (Van Rossum and Drake, 2009), with the use of the following libraries: PyTorch (Paszke et al., 2019); PyTorch Geometric (Fey and Lenssen, 2019); Torch Spatiotemporal (Cini and Marisca, 2022); Scikit-learn (Pedregosa et al., 2011); PyTorch Lightning (Falcon and The PyTorch Lightning team, 2019); Hydra (Yadan, 2019); Numpy (Harris et al., 2020). Below, we provide further details on the experiments conducted in Sec. 4 and report complete tables for both the MSE and mean absolute error (MAE) test metrics. Moreover, we provide additional information on the datasets in Tab. 6.

Table 6: Information on the datasets.

Dataset	Time series	Steps	Frequency	Domain
Weather	21	52695	10min	Weather
Solar-Energy	137	52559	10min	Energy
ECL	321	26303	Hourly	Electricity
Traffic	862	17543	Hourly	Transportation

Hyperparameter tuning For each experiment, we set a fixed batch size for each dataset. The hidden size is tuned between 32 and 256 for all datasets, with the addition of 16 for the Weather dataset. In Tab. 3, 12, 11, we used hyperparameters corresponding to the best configuration found during tuning. For Tab. 9, 10, Fig. 3, and 4, we used the same hyperparameters obtained from the tuning performed for Tab. 3, 11. Instead, in Tab. 7 and 8, we used fixed hyperparameters, identical for both sides of the comparison, without any tuning. The window size was set to 336 in Tab. 3, 9, and Fig. 3, except for the Solar dataset, for which it was set to 96. For the other tables, the window size was set to 96. Where not otherwise specified, the forecasting horizon is set to 96. Hyperparameters are fixed across horizons.

Empirical setup for D1: model configuration In Tab. 7, the global TimeMixer model is obtained by removing the learnable local parameters from the normalization module, while the global versions of both Crossformer and the reference Transformer are obtained by excluding their local embeddings. Analogously, the hybrid iTransformer is obtained by incorporating local embeddings into the input encoding.

Empirical setup for D2: preprocessing and exogenous variables In Tab. 8, covariates were removed from the reference Transformer and iTransformer, whereas they were included in the original implementations of PatchTST, DLinear, and Crossformer.

Empirical setup for D3: temporal processing In Tab. 3, we report the MSE and MAE performance of the reference architectures for various temporal processing modules, evaluated against baselines that do not include spatial processing operators. Tab. 9 presents the computational efficiency of the models for increasing horizons on the Electricity dataset. We employed the PyTorch Profiler (Paszke et al., 2019) to monitor GPU performance during training, specifically collecting the total CUDA execution time. Additionally, GPU memory usage was obtained using the PyG (Fey et al., 2025) function *get_gpu_memory_from_nvidia_smi*. To ensure a consistent evaluation, all measurements

Table 7: Comparison (MSE and MAE) of models with and without local parameters. Best average results are in **bold**.

D	Model	hybrid		global	
		MSE	MAE	MSE	MAE
Electr.	Transf.	**0.136**$_{\pm.000}$	**0.231**$_{\pm.000}$	0.151$_{\pm.000}$	0.242$_{\pm.000}$
	Crossformer	**0.141**$_{\pm.001}$	**0.235**$_{\pm.001}$	0.146$_{\pm.003}$	0.240$_{\pm.003}$
	TimeMixer	**0.151**$_{\pm.000}$	**0.248**$_{\pm.001}$	0.180$_{\pm.001}$	0.268$_{\pm.001}$
	iTransformer	**0.139**$_{\pm.000}$	**0.236**$_{\pm.001}$	0.154$_{\pm.000}$	0.245$_{\pm.000}$
Weather	Transf.	**0.153**$_{\pm.001}$	**0.198**$_{\pm.000}$	0.177$_{\pm.002}$	0.215$_{\pm.001}$
	Crossformer	**0.154**$_{\pm.003}$	0.225$_{\pm.004}$	0.164$_{\pm.003}$	**0.224**$_{\pm.007}$
	TimeMixer	**0.164**$_{\pm.002}$	**0.208**$_{\pm.001}$	0.178$_{\pm.001}$	0.216$_{\pm.001}$
	iTransformer	**0.154**$_{\pm.000}$	**0.199**$_{\pm.001}$	0.170$_{\pm.001}$	0.211$_{\pm.001}$
Traffic	Transf.	0.417$_{\pm.009}$	0.278$_{\pm.005}$	**0.392**$_{\pm.000}$	**0.260**$_{\pm.001}$
	Crossformer	0.540$_{\pm.014}$	0.279$_{\pm.007}$	**0.512**$_{\pm.007}$	**0.259**$_{\pm.004}$
	TimeMixer	0.464$_{\pm.001}$	0.328$_{\pm.003}$	**0.463**$_{\pm.001}$	**0.327**$_{\pm.003}$
	iTransformer	0.435$_{\pm.002}$	**0.275**$_{\pm.000}$	**0.409**$_{\pm.000}$	0.277$_{\pm.001}$
Solar	Transf.	**0.196**$_{\pm.000}$	**0.243**$_{\pm.001}$	0.205$_{\pm.001}$	0.247$_{\pm.002}$
	Crossformer	0.177$_{\pm.008}$	0.215$_{\pm.004}$	**0.166**$_{\pm.005}$	**0.204**$_{\pm.006}$
	TimeMixer	**0.366**$_{\pm.017}$	**0.396**$_{\pm.013}$	0.367$_{\pm.017}$	0.396$_{\pm.013}$
	iTransformer	**0.189**$_{\pm.001}$	**0.240**$_{\pm.004}$	0.197$_{\pm.002}$	0.243$_{\pm.001}$

Table 8: Comparison (MSE and MAE) of models with and without covariates. Best average results are in **bold**.

D	Model	w/ exog.		w/o exog.	
		MSE	MAE	MSE	MAE
Electr.	Transf.	**0.136**$_{\pm.000}$	**0.231**$_{\pm.000}$	0.155$_{\pm.001}$	0.247$_{\pm.000}$
	PatchTST	**0.128**$_{\pm.000}$	**0.222**$_{\pm.000}$	0.134$_{\pm.000}$	0.228$_{\pm.001}$
	Crossformer	**0.139**$_{\pm.002}$	**0.234**$_{\pm.003}$	0.141$_{\pm.001}$	0.235$_{\pm.001}$
	iTransformer	**0.154**$_{\pm.000}$	**0.245**$_{\pm.000}$	0.167$_{\pm.000}$	0.254$_{\pm.000}$
	DLinear	**0.193**$_{\pm.000}$	0.277$_{\pm.000}$	0.195$_{\pm.000}$	0.277$_{\pm.000}$
Weather	Transf.	**0.153**$_{\pm.001}$	**0.198**$_{\pm.000}$	0.161$_{\pm.000}$	0.208$_{\pm.001}$
	PatchTST	**0.174**$_{\pm.000}$	**0.213**$_{\pm.001}$	0.180$_{\pm.002}$	0.221$_{\pm.002}$
	Crossformer	0.154$_{\pm.003}$	0.225$_{\pm.002}$	0.154$_{\pm.003}$	0.225$_{\pm.004}$
	iTransformer	**0.170**$_{\pm.001}$	**0.211**$_{\pm.001}$	0.176$_{\pm.000}$	0.217$_{\pm.001}$
	DLinear	0.199$_{\pm.005}$	0.258$_{\pm.008}$	**0.196**$_{\pm.001}$	**0.248**$_{\pm.002}$
Traffic	Transf.	**0.417**$_{\pm.009}$	**0.278**$_{\pm.005}$	0.479$_{\pm.006}$	0.289$_{\pm.001}$
	PatchTST	**0.355**$_{\pm.000}$	**0.244**$_{\pm.000}$	0.383$_{\pm.001}$	0.261$_{\pm.001}$
	Crossformer	0.548$_{\pm.024}$	**0.278**$_{\pm.011}$	**0.540**$_{\pm.014}$	0.279$_{\pm.007}$
	iTransformer	**0.409**$_{\pm.000}$	**0.277**$_{\pm.001}$	0.444$_{\pm.001}$	0.290$_{\pm.001}$
	DLinear	**0.609**$_{\pm.000}$	**0.391**$_{\pm.000}$	0.648$_{\pm.000}$	0.395$_{\pm.000}$
Solar	Transf.	**0.196**$_{\pm.000}$	**0.243**$_{\pm.001}$	0.206$_{\pm.003}$	0.249$_{\pm.004}$
	PatchTST	**0.196**$_{\pm.001}$	**0.246**$_{\pm.004}$	0.225$_{\pm.003}$	0.268$_{\pm.003}$
	Crossformer	**0.176**$_{\pm.006}$	0.231$_{\pm.010}$	0.177$_{\pm.008}$	**0.215**$_{\pm.004}$
	iTransformer	**0.197**$_{\pm.002}$	**0.243**$_{\pm.001}$	0.221$_{\pm.003}$	0.256$_{\pm.002}$
	DLinear	**0.246**$_{\pm.001}$	**0.331**$_{\pm.000}$	0.285$_{\pm.001}$	0.372$_{\pm.001}$

related to model performance (Tab. 9 and 10) were conducted on the same machine running Oracle Linux Server 8.8, equipped with an Intel Xeon E5-2650 v3 CPU @ 2.30 GHz 20 (2 x 10) cores, 128 GB of system RAM, and an NVIDIA A100-PCIe GPU with 40 GB of HBM2 memory. Finally, we summarize these results in Fig. 3 which illustrate the trade-off between model performance and computational efficiency in terms of training batch time and GPU memory usage, on the Electricity dataset for a forecasting horizon of 96.

Table 9: Performance and resource utilization of the models selected in 3 on the Electricity dataset. Best performance is shown in **bold**, second best is underlined.

Model	Horizon	Batch Time (ms)	Batches per Second	GPU Mem. (MB)	CUDA Time (ms)
MLP	96	27.6±1.4	36.3±1.2	628.0	20.3
	192	27.6±1.4	36.3±1.2	628.0	27.1
	336	27.6±1.4	36.3±1.2	653.2	26.3
	720	27.6±1.4	36.3±1.2	705.6	31.8
RNN	96	36.7±0.9	28.2±0.6	3635.2	235.6
	192	37.3±2.9	27.8±1.8	3643.5	243.0
	336	37.3±2.9	27.8±1.8	3854.3	246.7
	720	37.3±2.9	27.8±1.8	3860.6	258.5
TCN	96	29.9±1.4	33.5±1.1	1217.3	62.0
	192	29.9±1.4	33.5±1.1	1217.3	82.0
	336	29.9±1.4	33.5±1.1	1219.4	85.1
	720	29.9±1.4	33.5±1.1	1240.4	87.8
Transf.	96	36.7±1.3	27.3±0.7	1942.9	119.3
	192	36.7±1.3	27.3±0.7	1961.7	144.0
	336	36.7±1.3	27.3±0.7	1959.7	148.1
	720	36.7±1.3	27.3±0.7	1976.4	164.7
Pyraf.	96	41.7±0.8	24.0±0.4	2559.4	189.9
	192	41.7±0.8	24.0±0.4	2561.5	191.9
	336	41.7±0.8	24.0±0.4	2563.6	192.8
	720	41.7±0.8	24.0±0.4	2567.8	198.7
DLinear	96	**18.9±1.1**	**52.9±1.7**	**615.5**	**10.7**
	192	**18.9±1.1**	**52.9±1.7**	**615.5**	**17.2**
	336	**18.9±1.1**	**52.9±1.7**	**638.5**	**19.1**
	720	**18.9±1.1**	**52.9±1.7**	**699.4**	**22.5**
PatchTST	96	34.3±0.5	29.1±0.4	1445.9	74.9
	192	34.3±0.5	29.1±0.4	1443.8	94.6
	336	34.3±0.5	29.1±0.4	1443.8	93.6
	720	34.3±0.5	29.1±0.4	1462.7	107.7
TimeMixer	96	97.6±93.5	10.9±0.7	2301.5	410.4
	192	97.6±93.5	10.9±0.7	2303.6	405.1
	336	97.6±93.5	10.9±0.7	2311.9	375.9
	720	97.6±93.5	10.9±0.7	2450.3	478.8

Empirical setup for D4: spatial processing In Tab. 11, we compare the reference architectures with baselines that include spatial processing. The reference architectures are employed with either an MLP or a pyramidal attention module for temporal processing, followed by a spatial attention module. The temporal modules were chosen for their advantageous trade-off between performance and computational efficiency (see Fig. 3 and Tab. 9). In Tab. 12, the iTransformer version without attention was obtained by replacing the spatial attention with a simple feedforward layer. Similarly to Fig. 3, Fig. 4 shows the trade-off between model performance and computational efficiency on the Electricity dataset for a forecasting horizon of 96. Since spatial processing often increases computational cost and reduces memory efficiency, we restrict the input window size to 96, equal to the forecasting horizon used in Tab. 11, 10, 12 and Fig. 4.

Table 10: Performance and resource utilization of the models selected in 4a on the Electricity dataset. Best performance is shown in **bold**, second best is underlined.

Model	Horizon	Batch Time (ms)	Batches per Second	GPU Mem. (MB)	CUDA Time (ms)
MLP + sp. att.	96	**40.2**$\pm$**1.0**	**24.9**$\pm$**0.5**	**1162.8**	**96.9**
	192	**40.6**$\pm$**1.3**	**24.7**$\pm$**0.6**	**1236.2**	**123.8**
	336	**40.6**$\pm$**1.3**	**24.7**$\pm$**0.6**	**1215.2**	**155.4**
	720	**40.6**$\pm$**1.3**	**24.7**$\pm$**0.6**	**1357.8**	**185.8**
Pyraf. + sp. att.	96	96.4$\pm$0.4	10.4$\pm$0.0	7822.9	783.6
	192	96.4$\pm$0.4	10.4$\pm$0.0	7908.8	787.3
	336	96.4$\pm$0.4	10.4$\pm$0.0	7837.5	808.8
	720	96.4$\pm$0.4	10.4$\pm$0.0	7940.3	861.9
iTransformer	96	50.8$\pm$1.3	19.7$\pm$0.4	1729.0	217.4
	192	50.7$\pm$1.1	19.7$\pm$0.4	1630.4	227.9
	336	50.7$\pm$1.1	19.7$\pm$0.4	1712.2	252.9
	720	50.7$\pm$1.1	19.7$\pm$0.4	1871.6	317.8
Crossformer	96	138.0$\pm$1.0	7.2$\pm$0.1	5702.8	912.8
	192	161.9$\pm$28.1	6.4$\pm$1.0	9412.4	1532.0
	336	161.9$\pm$28.1	6.4$\pm$1.0	16032.6	2516.0
	720	161.9$\pm$28.1	6.4$\pm$1.0	39527.4	5589.0
Modern TCN	96	145.4$\pm$1.1	6.9$\pm$0.0	10620.3	1967.0
	192	147.4$\pm$8.8	6.8$\pm$0.3	10620.3	1971.0
	336	147.4$\pm$8.8	6.8$\pm$0.3	10662.2	2005.0
	720	147.4$\pm$8.8	6.8$\pm$0.3	10857.2	2036.0

Table 11: Forecasting results (MSE and MAE) for a horizon of 96 steps for models *including* spatial processing. Best average results are in **bold**, second best are underlined.

Model	Electricity		Weather		Traffic		Solar	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
MLP + sp. attn.	0.140$\pm$.001	0.238$\pm$.001	0.157$\pm$.000	0.202$\pm$.001	0.435$\pm$.006	0.275$\pm$.001	0.201$\pm$.009	0.246$\pm$.003
Pyraf. + sp. attn.	0.139$\pm$.001	0.236$\pm$.001	0.157$\pm$.002	0.204$\pm$.001	**0.389**$\pm$**.002**	0.267$\pm$.001	0.188$\pm$.002	0.235$\pm$.003
iTransformer	0.148$\pm$.000	0.241$\pm$.000	0.171$\pm$.001	0.210$\pm$.001	0.393$\pm$.001	**0.266**$\pm$**.001**	0.208$\pm$.003	0.240$\pm$.006
Crosformer	**0.136**$\pm$**.000**	**0.232**$\pm$**.001**	**0.152**$\pm$**.003**	0.222$\pm$.004	0.527$\pm$.002	0.270$\pm$.003	**0.184**$\pm$**.008**	0.227$\pm$.006
ModernTCN	0.141$\pm$.000	0.237$\pm$.001	0.154$\pm$.001	**0.200**$\pm$**.001**	0.445$\pm$.001	0.287$\pm$.001	0.190$\pm$.001	**0.222**$\pm$**.002**

Table 12: Results (MSE and MAE) for iTransformer with or without space attention. Best average results are in **bold**.

Dataset	Space att.		Feedforward	
	MSE	MAE	MSE	MAE
Electricity	**0.148**$\pm$**.000**	0.241$\pm$.000	0.149$\pm$.001	**0.237**$\pm$**.001**
Weather	0.171$\pm$.001	0.210$\pm$.001	0.171$\pm$.000	0.210$\pm$.001
Traffic	0.393$\pm$.001	0.266$\pm$.001	**0.390**$\pm$**.001**	**0.258**$\pm$**.000**
Solar	0.208$\pm$.003	0.240$\pm$.006	**0.194**$\pm$**.001**	**0.230**$\pm$**.002**

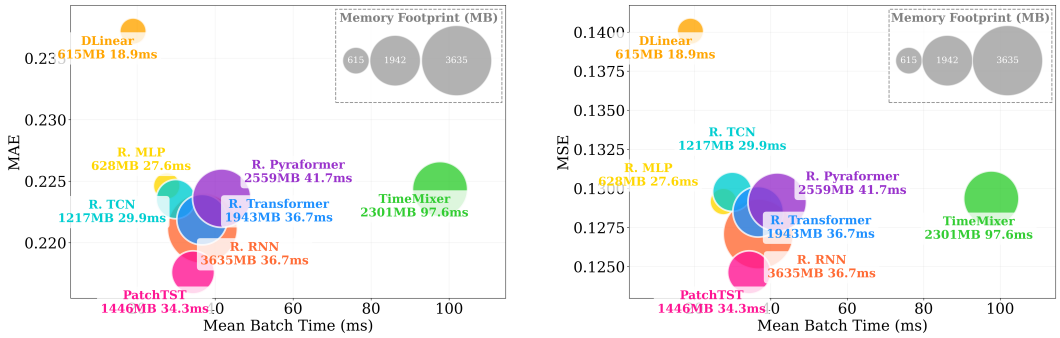


Figure 3: MAE and MSE performance versus mean batch time during training for models *not including* spatial processing, for a batch size of 512. Circle size indicates memory consumption.

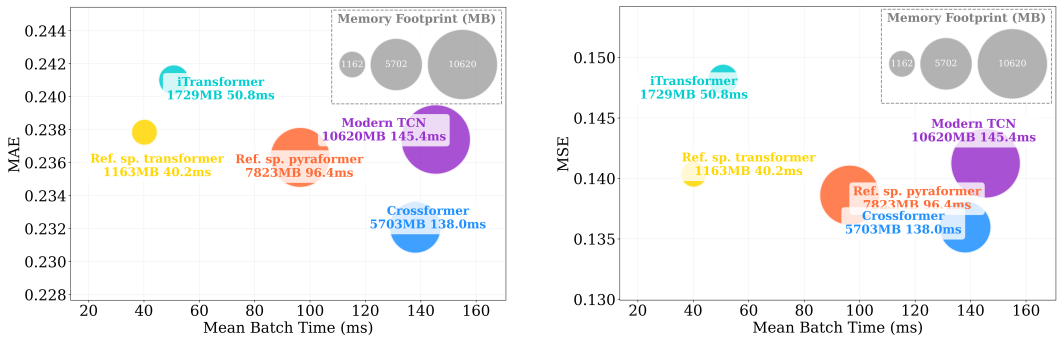


Figure 4: MAE and MSE performance versus mean batch time during training for models *including* spatial processing, for a batch size of 32. Circle size indicates memory consumption.

D Model Cards

In Tab. 13, we report an example of the usage of the newly introduced model cards for PatchTST.

E Additional Results

We present additional results covering a wider range of input windows, forecasting horizons, and model configurations.

Multi-horizon results Tab. 14 and Tab. 20 extend the results of Sec. 4.3 and Sec. 4.4 to a broader set of horizons (96, 192, 336, 720). We observe that increasing the forecasting horizon does not change the conclusions drawn in the corresponding sections, and results still show that—even for larger forecasting horizons—standard, streamlined architectures achieve performance comparable to current state-of-the-art models.

Extended ablation on iTransformer Tab. 15, Tab. 16, Tab. 19 and Tab. 17 expand the iTransformer ablation study of Tab. 12 to additional window sizes (96, 336, 720) and forecasting horizons (96, 192, 336, 720). These additional experiments further reinforce our analysis in Sec. 4.4, showing a general performance improvement when removing one of the core components introduced by state-of-the-art architectures.

Linear autoregressive models In Tab. 18, we evaluate the performance of two simple hybrid variants of Linear and DLinear against the local and global configurations. The hybrid DLinear variant is obtained by concatenating learnable local parameters to the input, whereas for Linear, we concatenate a one-hot encoding representing the time series. The results are consistent with the findings discussed in Sec. 4.1.

Table 13: Example of model cards for PatchTST on the Electricity dataset

FORECASTING MODEL CARD

Model setting

- *Window length*: fixed lookback window of 336
- *Transductive or inductive (cold start)*: inductive
- *Masking*: not applied/needed

D1. Model configuration

- *Global/local/hybrid*: global model
- *Hybrid parameters (non-shared)*: not applicable

D2. Preprocessing and exogenous variables

- *Scaling*: standard normalization (z-score) applied per series and in-batch RevInv normalization
- *Covariates/exogenous variables*: not used

D3. Temporal processing

- *Temporal modules*: convolutional encoding followed by patching-based Transformer layers
- *Complexity scaling with steps*: the time and space complexity scales quadratically with the number of patches (self-attention)

D4. Spatial processing

- *Spatial modules*: not applicable
- *Complexity scaling with nodes*: not applicable

F LARGE LANGUAGE MODELS

We acknowledge the use of Large Language Models to assist in polishing the manuscript by making minor edits to single sentences.

G CODE OF ETHICS

The work presented in this paper is about basic machine learning research, and all experiments are conducted on standard, publicly available datasets. The authors have read and adhere to the ICLR Code of Ethics and do not foresee any direct ethical concerns or potential for misuse.

Table 14: Results (MSE and MAE) for multiple horizons. Best mean results are in **bold**, second best are underlined.

Model	H	Electricity		Weather		Traffic		Solar	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
OLS Global	96	0.140	0.237	0.174	0.234	0.410	0.282	0.222	0.291
	192	0.154	0.250	0.215	0.272	0.423	0.288	0.249	0.309
	336	0.169	0.268	0.260	0.309	0.436	0.295	0.269	0.324
	720	0.204	0.301	0.323	0.361	0.466	0.315	0.271	0.327
OLS Local	96	0.134	0.230	**0.144**	0.209	0.426	0.298	0.223	0.295
	192	0.149	0.245	**0.187**	0.254	0.438	0.304	0.251	0.313
	336	0.165	0.263	**0.240**	0.298	0.452	0.312	0.270	0.327
	720	0.201	0.297	**0.316**	0.358	0.482	0.330	0.272	0.331
R. MLP	96	0.129±0.000	0.225±0.000	0.148±0.001	0.198±0.000	0.376±0.000	0.253±0.001	0.194±0.003	0.239±0.002
	192	0.149±0.000	0.245±0.001	0.191±0.001	0.241±0.000	0.403±0.001	0.270±0.001	0.227±0.001	0.262±0.001
	336	0.166±0.000	0.262±0.000	0.245±0.001	0.281±0.001	0.417±0.001	0.277±0.001	0.249±0.001	0.277±0.002
	720	0.205±0.000	0.296±0.000	0.324±0.002	0.338±0.001	0.456±0.001	0.295±0.001	0.254±0.000	0.279±0.001
R. RNN	96	0.127±0.001	0.221±0.000	0.148±0.000	0.199±0.001	0.362±0.002	0.248±0.001	0.192±0.002	**0.236±0.001**
	192	0.167±0.001	0.267±0.001	0.190±0.001	0.244±0.001	0.411±0.005	0.282±0.005	0.230±0.003	0.270±0.002
	336	0.186±0.001	0.287±0.001	0.244±0.001	0.286±0.001	0.423±0.019	0.285±0.005	0.251±0.007	0.284±0.007
	720	0.225±0.002	0.323±0.001	0.324±0.003	0.342±0.003	0.497±0.040	0.297±0.001	0.249±0.005	0.281±0.005
R. TCN	96	0.130±0.000	0.224±0.000	0.148±0.000	0.200±0.001	0.364±0.003	0.253±0.002	0.193±0.004	0.243±0.005
	192	0.148±0.000	0.240±0.000	0.195±0.001	0.246±0.000	0.382±0.001	0.261±0.001	**0.221±0.002**	**0.253±0.002**
	336	0.165±0.001	0.258±0.001	0.252±0.002	0.289±0.001	0.397±0.002	0.274±0.005	0.249±0.005	**0.271±0.005**
	720	0.202±0.002	0.292±0.001	0.329±0.001	0.342±0.001	**0.434±0.001**	0.291±0.005	0.246±0.004	0.273±0.003
R. Transf.	96	0.129±0.001	0.222±0.001	0.149±0.001	0.203±0.002	0.362±0.003	0.249±0.002	0.203±0.006	0.245±0.002
	192	0.146±0.000	0.238±0.000	0.198±0.001	0.250±0.002	**0.374±0.001**	**0.255±0.001**	0.224±0.001	0.260±0.001
	336	0.164±0.001	0.258±0.001	0.248±0.001	0.289±0.002	**0.393±0.002**	0.271±0.007	0.245±0.004	**0.271±0.003**
	720	0.202±0.003	0.294±0.002	0.330±0.010	0.344±0.006	**0.434±0.004**	0.294±0.008	0.248±0.004	0.277±0.004
R. Pyraf.	96	0.129±0.001	0.224±0.001	0.148±0.001	0.199±0.001	0.365±0.002	0.251±0.003	**0.189±0.003**	**0.236±0.004**
	192	0.147±0.001	0.240±0.000	0.196±0.003	0.246±0.002	0.384±0.003	0.262±0.004	0.224±0.001	0.256±0.001
	336	0.164±0.001	0.258±0.000	0.248±0.003	0.287±0.003	0.397±0.002	0.269±0.002	**0.244±0.001**	**0.271±0.002**
	720	0.200±0.000	0.293±0.000	0.328±0.003	0.344±0.002	**0.434±0.002**	0.297±0.003	0.248±0.002	**0.272±0.001**
TimeMixer	96	0.129±0.001	0.224±0.000	0.147±0.001	0.197±0.000	0.373±0.002	0.271±0.003	0.199±0.001	0.245±0.000
	192	0.147±0.001	0.241±0.000	0.191±0.000	**0.239±0.000**	0.396±0.001	0.283±0.001	0.230±0.003	0.268±0.003
	336	0.166±0.001	0.260±0.001	0.244±0.002	0.281±0.002	0.416±0.001	0.297±0.001	**0.244±0.002**	0.280±0.000
	720	0.206±0.003	0.297±0.003	0.321±0.003	0.334±0.003	0.450±0.006	0.318±0.003	**0.245±0.004**	0.280±0.001
PatchTST	96	**0.125±0.000**	**0.218±0.000**	0.148±0.001	**0.195±0.001**	**0.345±0.000**	**0.234±0.000**	0.197±0.001	0.244±0.004
	192	**0.143±0.000**	**0.236±0.000**	0.194±0.000	**0.239±0.001**	0.384±0.001	0.258±0.001	0.227±0.002	0.260±0.002
	336	**0.160±0.000**	**0.254±0.000**	0.247±0.000	**0.279±0.001**	0.396±0.001	**0.264±0.000**	0.249±0.001	0.273±0.002
	720	**0.197±0.001**	**0.288±0.001**	0.322±0.002	**0.333±0.001**	0.435±0.001	**0.286±0.000**	0.247±0.001	0.273±0.002
DLinear	96	0.140±0.000	0.237±0.000	0.173±0.000	0.232±0.001	0.407±0.000	0.283±0.000	0.246±0.001	0.331±0.000
	192	0.154±0.000	0.250±0.001	0.216±0.001	0.274±0.003	0.421±0.000	0.290±0.000	0.267±0.001	0.342±0.000
	336	0.169±0.000	0.268±0.000	0.265±0.002	0.318±0.003	0.433±0.000	0.296±0.000	0.289±0.001	0.353±0.000
	720	0.203±0.000	0.300±0.000	0.331±0.002	0.373±0.003	0.461±0.000	0.314±0.000	0.294±0.001	0.355±0.000

Table 15: Ablation study on itransformer for window=96 across different forecasting horizons. Best mean results are in **bold**.

Dataset	Horizon	With space attention		Without space attention	
		MSE	MAE	MSE	MAE
Electricity	96	**0.148**±**0.000**	0.241±0.000	0.149±0.001	**0.237**±**0.001**
	192	0.166±0.001	0.259±0.001	**0.161**±**0.000**	**0.250**±**0.000**
	336	0.179±0.002	0.274±0.002	0.179±0.000	**0.268**±**0.000**
	720	**0.218**±**0.000**	0.309±0.001	0.219±0.000	**0.303**±**0.000**
Weather	96	0.171±0.001	0.210±0.001	0.171±0.000	0.210±0.001
	192	0.221±0.001	0.255±0.000	**0.219**±**0.001**	**0.253**±**0.001**
	336	0.276±0.000	0.295±0.000	**0.275**±**0.000**	**0.294**±**0.000**
	720	0.353±0.000	0.346±0.000	**0.352**±**0.001**	**0.345**±**0.000**
Traffic	96	0.393±0.001	0.266±0.001	**0.390**±**0.001**	**0.258**±**0.000**
	192	0.424±0.001	0.286±0.001	**0.409**±**0.000**	**0.268**±**0.000**
	336	0.430±0.001	0.289±0.001	**0.423**±**0.000**	**0.274**±**0.000**
	720	0.455±0.001	0.303±0.000	**0.454**±**0.000**	**0.292**±**0.000**
Solar	96	0.208±0.003	0.240±0.006	**0.194**±**0.001**	**0.230**±**0.002**
	192	0.233±0.003	0.257±0.003	**0.226**±**0.001**	0.257±0.003
	336	**0.244**±**0.000**	0.273±0.002	0.245±0.003	**0.267**±**0.002**
	720	0.255±0.002	0.280±0.001	**0.249**±**0.001**	**0.271**±**0.001**

Table 16: Ablation study on itransformer for window=336 across different forecasting horizons. Best mean results are in **bold**.

Dataset	Horizon	With space attention		Without space attention	
		MSE	MAE	MSE	MAE
Electricity	96	0.135±0.001	0.229±0.001	**0.130**±**0.000**	**0.223**±**0.000**
	192	0.155±0.001	0.248±0.001	**0.149**±**0.000**	**0.242**±**0.000**
	336	0.172±0.002	0.267±0.000	**0.166**±**0.000**	**0.260**±**0.000**
	720	**0.201**±**0.002**	0.294±0.002	0.205±0.000	0.294±0.000
Weather	96	0.159±0.001	0.207±0.000	**0.153**±**0.000**	**0.202**±**0.001**
	192	0.203±0.001	0.249±0.001	**0.197**±**0.001**	**0.245**±**0.002**
	336	0.252±0.002	0.286±0.000	**0.249**±**0.001**	**0.284**±**0.001**
	720	**0.326**±**0.003**	**0.338**±**0.002**	0.328±0.002	0.339±0.001
Traffic	96	0.363±0.000	0.257±0.001	**0.359**±**0.001**	**0.247**±**0.001**
	192	0.385±0.002	0.269±0.001	**0.377**±**0.001**	**0.256**±**0.001**
	336	0.397±0.002	0.277±0.001	**0.386**±**0.000**	**0.261**±**0.000**
	720	0.423±0.001	0.291±0.000	**0.420**±**0.001**	**0.282**±**0.000**
Solar	96	0.195±0.000	0.252±0.003	**0.189**±**0.001**	**0.232**±**0.001**
	192	0.222±0.004	0.269±0.001	**0.207**±**0.000**	**0.249**±**0.000**
	336	0.230±0.007	0.276±0.004	**0.214**±**0.000**	**0.255**±**0.001**
	720	0.223±0.002	0.274±0.003	**0.216**±**0.003**	**0.258**±**0.002**

Table 17: Ablation study on itransformer for horizon=96 across different window lengths. Best mean results are in **bold**.

Dataset	Window	With space attention		Without space attention	
		MSE	MAE	MSE	MAE
Electricity	96	**0.148**±0.000	0.241±0.000	0.149±0.001	**0.237**±0.001
	336	0.135±0.001	0.229±0.001	**0.130**±0.000	**0.223**±0.000
	720	0.135±0.002	0.231±0.001	**0.132**±0.000	**0.227**±0.001
Weather	96	0.171±0.001	0.210±0.001	0.171±0.000	0.210±0.001
	336	0.159±0.001	0.207±0.000	**0.153**±0.000	**0.202**±0.001
	720	0.155±0.002	0.208±0.002	**0.149**±0.000	**0.201**±0.001
Traffic	96	0.393±0.001	0.266±0.001	**0.390**±0.001	**0.258**±0.000
	336	0.363±0.000	0.257±0.001	**0.359**±0.001	**0.247**±0.001
	720	**0.353**±0.004	0.256±0.001	0.357±0.000	**0.251**±0.000
Solar	96	0.208±0.003	0.240±0.006	**0.194**±0.001	**0.230**±0.002
	336	0.195±0.000	0.252±0.003	**0.189**±0.001	**0.232**±0.001
	720	**0.175**±0.003	0.245±0.004	0.182±0.008	**0.235**±0.006

Table 18: Comparison (MSE and MAE) of Linear models in their local, global, and hybrid variants. A dash (–) marks experiments beyond our computational budget. Best mean results are in **bold**.

Dataset	Model	Hybrid		Global		Local	
		MSE	MAE	MSE	MAE	MSE	MAE
Electr.	DLinear	0.195±.000	0.277±.000	0.195±.000	0.277±.000	**0.184**±.000	**0.270**±.000
	OLS	0.194±.000	0.277±.000	0.195±.000	0.277±.000	**0.184**±.000	**0.270**±.000
Weather	DLinear	0.198±.001	0.254±.002	0.196±.001	0.248±.002	**0.161**±.001	**0.233**±.001
	OLS	0.196±.000	0.254±.000	0.195±.000	0.253±.000	**0.161**±.000	**0.233**±.000
Traffic	DLinear	0.648±.000	**0.395**±.000	0.648±.000	**0.395**±.000	**0.647**±.000	0.403±.000
	OLS	-	-	0.649±.000	**0.396**±.000	**0.647**±.000	0.403±.000
Solar	DLinear	0.286±.000	0.375±.001	**0.285**±.001	**0.372**±.001	0.286±.000	0.376±.001
	OLS	**0.285**±.000	**0.372**±.000	**0.285**±.000	**0.372**±.000	0.286±.000	0.374±.000

Table 19: Ablation study on itransformer for window=720 across different forecasting horizons. Best mean results are in **bold**.

Dataset	Horizon	With space attention		Without space attention	
		MSE	MAE	MSE	MAE
Electricity	96	0.135±0.002	0.231±0.001	**0.132±0.000**	**0.227±0.001**
	192	0.157±0.001	0.252±0.001	**0.151±0.001**	**0.245±0.001**
	336	0.175±0.001	0.272±0.001	**0.165±0.000**	**0.263±0.000**
	720	**0.195±0.001**	**0.289±0.002**	0.201±0.001	0.294±0.001
Weather	96	0.155±0.002	0.208±0.002	**0.149±0.000**	**0.201±0.001**
	192	0.202±0.002	0.250±0.001	**0.195±0.001**	**0.247±0.002**
	336	0.249±0.002	**0.288±0.002**	0.249±0.001	0.289±0.001
	720	0.322±0.002	0.342±0.000	**0.317±0.001**	**0.337±0.001**
Traffic	96	**0.353±0.004**	0.256±0.001	0.357±0.000	**0.251±0.000**
	192	0.372±0.002	0.267±0.000	**0.369±0.000**	**0.260±0.000**
	336	0.388±0.002	0.276±0.001	**0.383±0.001**	**0.269±0.001**
	720	**0.417±0.002**	0.290±0.001	0.420±0.000	**0.287±0.000**
Solar	96	**0.175±0.003**	0.245±0.004	0.182±0.008	**0.235±0.006**
	192	**0.197±0.001**	0.262±0.002	0.200±0.003	**0.258±0.003**
	336	0.211±0.002	0.272±0.003	**0.209±0.002**	**0.262±0.002**
	720	0.216±0.001	0.275±0.003	**0.210±0.000**	**0.264±0.001**

Table 20: Results (MSE and MAE) across datasets and horizons. Best mean results are in **bold**, second best are underlined. A dash (–) marks experiments beyond our computational budget.

Model	H	Electricity		Weather		Traffic		Solar	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
MLP + sp. attn.	96	0.140±0.001	0.238±0.001	0.157±0.000	0.202±0.001	0.435±0.006	0.275±0.001	0.201±0.009	0.246±0.003
	192	0.167±0.002	0.263±0.002	0.206±0.001	**0.249±0.001**	0.454±0.008	0.286±0.003	0.237±0.003	0.272±0.003
	336	0.182±0.002	0.281±0.002	0.264±0.001	0.292±0.001	0.472±0.007	0.292±0.004	0.266±0.006	0.294±0.005
	720	0.209±0.008	0.305±0.006	0.349±0.001	0.345±0.001	0.523±0.003	0.316±0.003	0.266±0.004	0.304±0.007
Pyraf. + sp. attn.	96	0.139±0.001	0.236±0.001	0.157±0.002	0.204±0.001	**0.389±0.002**	0.267±0.001	0.188±0.002	0.235±0.003
	192	0.157±0.000	0.254±0.000	0.207±0.001	0.251±0.002	**0.410±0.001**	**0.276±0.002**	0.234±0.003	0.270±0.008
	336	0.174±0.002	0.273±0.002	0.267±0.001	0.294±0.002	**0.420±0.001**	**0.283±0.001**	0.248±0.001	0.281±0.004
	720	**0.194±0.001**	**0.292±0.001**	0.348±0.002	0.347±0.000	**0.449±0.002**	**0.300±0.001**	0.249±0.001	0.280±0.002
iTransformer	96	0.148±0.000	0.241±0.000	0.171±0.001	0.210±0.001	0.393±0.001	**0.266±0.001**	0.208±0.003	0.240±0.006
	192	0.166±0.001	0.259±0.001	0.221±0.001	0.255±0.000	0.424±0.001	0.286±0.001	0.233±0.003	0.257±0.003
	336	0.179±0.002	0.274±0.002	0.276±0.000	0.295±0.000	0.430±0.001	0.289±0.001	0.244±0.000	0.273±0.002
	720	0.218±0.000	0.309±0.001	0.353±0.000	0.346±0.000	0.455±0.001	0.303±0.000	0.255±0.002	0.280±0.001
Crosformer	96	**0.136±0.000**	**0.232±0.001**	**0.152±0.003**	0.222±0.004	0.527±0.002	0.270±0.003	**0.184±0.008**	0.227±0.006
	192	0.161±0.003	0.257±0.003	**0.200±0.007**	0.271±0.007	0.541±0.002	**0.276±0.002**	**0.207±0.015**	**0.246±0.003**
	336	0.187±0.007	0.286±0.007	0.263±0.001	0.325±0.003	0.561±0.005	0.289±0.002	**0.222±0.016**	**0.252±0.001**
	720	–	–	0.359±0.007	0.389±0.004	–	–	**0.214±0.001**	**0.250±0.004**
ModernTCN	96	0.141±0.000	0.237±0.001	0.154±0.001	**0.200±0.001**	0.445±0.001	0.287±0.001	0.190±0.001	**0.222±0.002**
	192	**0.156±0.001**	**0.249±0.001**	0.201±0.002	0.252±0.002	–	–	0.235±0.001	0.250±0.002
	336	**0.172±0.003**	**0.263±0.001**	**0.254±0.003**	**0.291±0.002**	–	–	0.258±0.004	0.266±0.003
	720	0.201±0.000	**0.292±0.000**	**0.338±0.005**	**0.343±0.005**	–	–	0.262±0.003	0.277±0.004