Why Do Time Series Models Need Long Context Windows?

Luca Butera
Università della Svizzera Italiana, IDSIA
luca.butera@usi.ch

Giovanni De Felice
Università della Svizzera Italiana

Andrea Cini
EPFL

Cesare Alippi
Università della Svizzera Italiana, IDSIA
Politecnico di Milano

Abstract

Modern deep learning models for forecasting *groups of time series* rely on increasingly longer observation windows. However, the benefit of increasing the window size is often simply attributed to capturing long-range dependencies, and broader discussion on how *global* forecasting models leverage input observations has been limited. In this paper, we show that forecasting groups of time series involves two objectives: (i) *generative process identification* (GPI), i.e., inferring the specific process generating the input sequence, and (ii) *conditional forecasting* (CF), i.e., predicting future values given input observations. From this perspective, optimal predictions can be interpreted as an average over plausible data-generating processes, weighted by their likelihood given the input window. This suggests another explanation for the benefits of long context windows: they reduce the uncertainty about which specific process is generating the input time series during operation. We prove that even for processes with memory length P, an input window size strictly larger than P is *necessary* to achieve the minimum attainable error. Finally, we show how decoupling GPI and CF can improve computational scalability without compromising accuracy. Experiments on synthetic and real-world data validate our insights and their relevance for designing forecasting architectures.

1 Introduction

Deep learning approaches have become primary tools in forecasting large collections of time series [2, 3]. While the standard statistical approach is to independently fit a *local* model for each target time series [4, 5], more recent advancements show that a single, shared, *global* model can be effective across multiple series [6, 7, 8, 9]. In this context, recent research has focused on architectures capable of handling increasingly longer observation windows [10, 11, 12, 13], typically attributing performance improvements to the ability to capture long-range dependencies. While global models have been used *transductively* to forecast future observations of the training time series, several time series *foundation* models have been

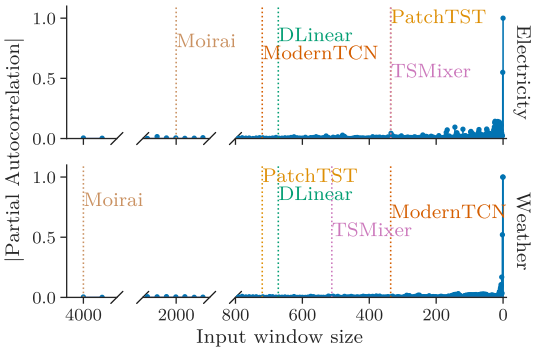


Figure 1: Partial autocorrelation at different time lags in two widely used forecasting benchmarks [1]. Vertical lines indicate the best window length for popular forecasting models.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

recently introduced [14, 15, 16, 17]. These are global models that aim to *inductively* forecast previously unseen time series. For this purpose, they are trained on vast corpora of time series coming from different domains, and usually rely on very long input windows and high model complexity. Interestingly, as Fig. 1 illustrates, in recent forecasting architectures, the window size selected with hyperparameter tuning is considerably larger than the number of time lags one would consider relevant, e.g., by looking at autocorrelation. Indeed, Montero-Manso and Hyndman [9] observed that transductive global models inherently require longer windows and higher capacity than local counterparts to guarantee the same performance. Beyond time series forecasting, there have been extensive efforts to understand how models trained on multiple tasks transfer to a new one given sufficient context [18, 19, 20, 21], a capability known as *in-context learning* (ICL) [22]. Nonetheless, discussion on how global forecasting models leverage input data beyond modeling temporal dependencies has been limited.

In this paper, we aim to fill this void and show that training a model to minimize forecasting error over groups of time series can be seen as involving two different tasks: (i) *generative process identification* (GPI), i.e., inferring the generative process that produced the input observations, and (ii) *conditional forecasting* (CF), i.e., predicting the next observations conditioned on the past. While most architectures do not explicitly decouple these tasks, as we will show, they can be viewed as inherent in how global models are trained and used in practice, regardless of whether they are used *transductively* or *inductively*. We show that optimal predictions can be interpreted as a *posterior average* over the possible generative processes, i.e., an average of forecasts for all plausible processes, each weighted by the estimated likelihood of that process generating the input sequence. This has important theoretical and methodological implications. Our main novel contributions and findings are as follows.

- **Problem formulation.** We formulate the task of learning a global forecasting model as learning from data generated by multiple generative processes from a common parametric family (Sec. 2). This formulation provides a framework for characterizing time series foundation models and understanding the associated learning problem.
- **The role and cost of GPI.** We provide theoretical and empirical evidence showing that, especially for foundation models, the input length and model capacity required for accurate forecasts are driven by GPI. We show that, in order to address GPI, global models may require an input window length *strictly larger* than that which would be needed to forecast a single time series (Sec. 3.2). Moreover, higher uncertainty about which specific process is generating the input time series must be mitigated by increasing the window size. We show how, in foundation models, this can induce worse performance-cost trade-offs w.r.t. domain-specific global models (Sec. 3.3).
- **Methodology.** We show that decoupling GPI and CF enables novel designs that can, e.g., improve scalability by amortizing the cost of GPI through dedicated model components (Sec. 4).

We support our analysis with theoretical results and empirical evidence on both synthetic and real-world datasets. Ultimately, we argue that in time series foundation models, ICL can be seen as leveraging the input window to reduce uncertainty about which process generated the data. We believe our findings can inform the design of the next generation of time series foundation models, leading to more efficient and principled approaches that exploit the specifics of the GPI and CF tasks. The paper is organized as follows: Sec. 2 introduces the proposed problem formulation; Sec. 3 introduces the GPI/CF perspective, provides theoretical and empirical evidence in its support, and demonstrates its impact on global and foundation models; Sec. 4 assesses a methodology to incorporate our insights in model design.

2 Problem formulation

We begin by defining the problem setting and the associated system model. Note that, throughout the paper, with a slight abuse of notation, we use the same symbols to denote both random variables and their realizations.

Problem setting. Consider a set $\mathcal{C}$ of N time series,

$$\mathcal{C} = \{\boldsymbol{x}^i_{0:T^i} \mid i \in [1, N]\}$$

where the i-th time series $\boldsymbol{x}^i_{0:T^i} \coloneqq \left(x^i_0, x^i_1, \ldots, x^i_{T^i-1}\right)$ consists of a sequence of T^i real-valued univariate observations. The set $\mathcal{C}$ can be *heterogeneous*, i.e., can contain time series from different

application domains (e.g., transportation, energy, climatology) and measuring different quantities. We focus on the univariate case for simplicity; the formulation can be extended to the multivariate case by considering architectures, e.g., [23], that can handle an arbitrary number of input channels.

Time series forecasting. The objective is to learn a parametric model $F(\cdot; \boldsymbol{\Theta})$ that, given a window of observations $\boldsymbol{x}^n_{t-W:t}$ ($W \geq 1$), predicts H future values $\boldsymbol{x}^n_{t:t+H}$ as

$$\hat{\boldsymbol{x}}^n_{t:t+H} = F(\boldsymbol{x}^n_{t-W:t};\, \boldsymbol{\Theta}), \tag{1}$$

where $\hat{\boldsymbol{x}}^n_{t:t+H}$ denotes predictions, and $\boldsymbol{\Theta}$ are learnable parameters. We use the superscript n to denote a generic target time series and cover both the transductive ($\boldsymbol{x}^n_{0:T^n} \in \mathcal{C}$) and inductive ($\boldsymbol{x}^n_{0:T^n} \notin \mathcal{C}$) scenarios. Note that the latter setting is typical of foundation models. Given a family of models, the objective is then to find the parameters' values that minimize the expected point-prediction loss

$$\{\boldsymbol{\Theta}_{opt}\} = \arg\min_{\boldsymbol{\Theta}} \mathcal{L}_W(\boldsymbol{\Theta}) = \arg\min_{\boldsymbol{\Theta}} \mathbb{E}\big[\ell\left(\boldsymbol{x}^n_{t:t+H}, F\left(\boldsymbol{x}^n_{t-W:t}; \boldsymbol{\Theta}\right)\right)\big] \tag{2}$$

between the target $\boldsymbol{x}^n_{t:t+H}$ and the prediction, as measured by loss function $\ell(\cdot)$, e.g., *mean squared error* (MSE) or *mean absolute error* (MAE). In the following, we focus on *global* forecasting models, i.e., models that are trained on data from all available N time series, as opposed to *local* models, which are trained on a single time series. Foundation models can be seen as a special case of global models specifically tailored to the inductive setting.

System model. While alternative formulations are possible, we use one that covers a wide range of scenarios. We assume that the generic n-th target time series is generated by a process belonging to a family of stochastic processes $\mathcal{G}$ characterized by a parameter vector $\boldsymbol{\phi}^n$. In particular, we assume the n-th time series is generated according to

$$x^n_t \sim \mathbb{P}_x(x^n_t | \boldsymbol{x}^n_{<t}; \boldsymbol{\phi}^n)\,, \quad \text{where} \quad \boldsymbol{\phi}^n \sim \boldsymbol{\mathcal{P}}_{\boldsymbol{\phi}}, \quad \forall t. \tag{3}$$

Here, $\boldsymbol{x}^n_{<t} \coloneqq \left(x^n_{t-1}, x^n_{t-2}, \dots\right)$ and $\mathbb{P}_x(\,\cdot\,; \boldsymbol{\phi}^n)$ is a time-invariant parametric distribution. $\boldsymbol{\mathcal{P}}_{\boldsymbol{\phi}}$ denotes the time-invariant target distribution over possible process parameters $\boldsymbol{\phi}^n$, defined on a d_p-manifold $\mathbb{M}^{d_p}$. The processes generating the time series in $\mathcal{C}$ are then associated with the set of i.i.d. random variables $\{\boldsymbol{\phi}^i\}_{i=1}^N$ distributed as $\boldsymbol{\mathcal{P}}_{\boldsymbol{\phi}}$. Note that $\mathbb{P}_x(\cdot)$ and $\boldsymbol{\mathcal{P}}_{\boldsymbol{\phi}}$ are shared across all processes. This system model accounts for the multitude of different dynamics that a global model aims to approximate. For example, we can consider different regions of $\mathbb{M}^{d_p}$ to correspond to different domains (e.g., meteorology, transportation, utilities). For the data typically used to train foundation models, we can expect $\boldsymbol{\mathcal{P}}_{\boldsymbol{\phi}}$ to have high variance and multiple modes, e.g., one for each target domain. While for domain-specific applications, one could expect $\boldsymbol{\mathcal{P}}_{\boldsymbol{\phi}}$ to have a smaller variance and result in time series with similar dynamics.

3 The role of process identification

In this section, we analyze how a global model can leverage the input observations to produce accurate forecasts across different time series. To simplify the notation, we consider 1-step-ahead forecasting.

3.1 Process identification and conditional forecasting

Given the task defined in Sec. 2, the objective is often to obtain predictions such that

$$\hat{\boldsymbol{x}}^n_t \approx \mathbb{E}\left[x^n_t \,\middle|\, \boldsymbol{x}^n_{t-W:t}\right], \tag{4}$$

choosing the MSE for the loss function $\ell\left(\cdot\right)$ (Eq. 2) [24]. Recall that each time series might be generated by a different stochastic process belonging to the family in Eq. 3. The expectation in Eq. 4 can be rewritten using the law of total conditional expectation as

$$\mathbb{E}\big[\mathbb{E}\big[x^n_t \big|\, \boldsymbol{x}^n_{t-W:t}, \boldsymbol{\phi}^n\big] \big|\, \boldsymbol{x}^n_{t-W:t}\big] = \int_{\mathbb{M}^{d_p}} \underbrace{\mathbb{E}\left[x^n_t \,\middle|\, \boldsymbol{x}^n_{t-W:t}, \boldsymbol{\phi}^n\right]}_{\text{CF}} \underbrace{p\left(\boldsymbol{\phi}^n \,\middle|\, \boldsymbol{x}^n_{t-W:t}\right)}_{\text{GPI}} d\boldsymbol{\phi}^n. \tag{5}$$

Eq. 5 shows that optimal predictions can be seen as a *Bayesian posterior average* [25] over the generating processes parameters. This highlights two coupled aspects of the forecasting problem:

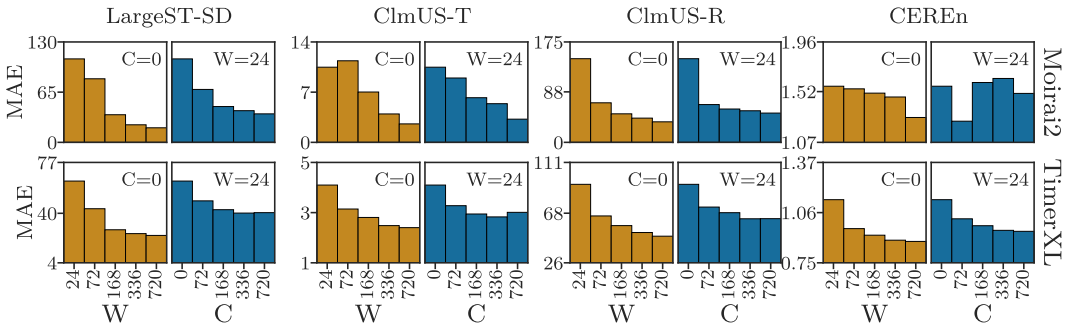


Figure 2: Pre-trained foundation models MAE, $H = 24$. For each pair of subplots, the left histogram (orange) shows the error for the *contiguous* setting (increasing W, $C = 0$), while the right histogram (blue) shows the error for the *disconnected* setting (minimum W, increasing C).

- ***Generative process identification*** **(GPI):** the task of inferring the data-generating process, captured by the posterior $p\left(\boldsymbol{\phi}^n \mid \boldsymbol{x}^n_{t-W:t}\right)$, which quantifies the likelihood of each parameter set $\boldsymbol{\phi}^n$ given the observed sequence $\boldsymbol{x}^n_{t-W:t}$.
- ***Conditional forecasting*** **(CF):** the task of predicting the next observation conditioned on the past, captured by the conditional expectation $\mathbb{E}\left[x^n_t \mid \boldsymbol{x}^n_{t-W:t}, \boldsymbol{\phi}^n\right]$.

Looking at Eq. 5, we see how input information relates to CF and GPI. The entire input sequence would be useful for GPI, as it can help contract the posterior $p\left(\boldsymbol{\phi}^n \mid \cdot\right)$. Conversely, for the CF task, part of the input might be redundant. Consider the case where, given $\boldsymbol{\phi}^n$, x^n_t depends only on the most recent $w < W$ observations, where W is the model's input window length. This suggests that GPI can increase the number of input observations needed for accurate forecasts beyond that which would be required by CF alone. Moreover, CF exploits the temporal dependencies between the input $\boldsymbol{x}^n_{t-W:t}$ and the target x^n_t, thus, the temporal distance between the two is relevant. In contrast, GPI relies on the relationship between observations and the process generating them; consequently, (e.g., if the process is stationary), it can be based on sequences non-adjacent to x^n_t. We can interpret GPI as a form of ICL from sequences of observations [20]. Indeed, recent works on language modeling view ICL as an implicit Bayesian inference algorithm, where the model infers a latent variable characterizing the target task from input/output examples [18, 21]. Under this interpretation, global time series models perform ICL by leveraging the input time series to infer the target task (i.e., to perform GPI). This links the window length W to the number of ICL examples. In standard ICL, performance improves as this number increases [26, 27]. Here, longer input windows can improve forecasting accuracy by reducing uncertainty about which process generated the data.

Evidence in foundation models Under the model in Eq. 3, GPI and CF are inherent to the problem of learning global models for groups of time series, and we would expect existing pre-trained foundation models to behave as if implicitly addressing both. To assess this, we probe two recent foundation models, Moirai2 [28] and TimerXL [29], comparing the effect of increasing the input length under two settings: (i) the standard *contiguous* setting, where we feed the model a *window* of length W, $\boldsymbol{x}^n_{t-W:t}$, for increasing values of $W \geq W_{min}$; (ii) a *disconnected* setting, where the window length is fixed to $W = W_{min}$ and we increase the input length by prepending an older *context* of length C, for increasing values of C, forming the concatenation $\left[\boldsymbol{x}^n_{t'-C:t'} \| \boldsymbol{x}^n_{t-W:t}\right]$ with $t' = t - W - S$. Here S denotes the temporal gap between the window and the additional context. Crucially, concatenation leaves the model blind to this gap. The rationale is that, if these models' performance is mainly due to capturing long-range temporal dependencies for CF, increasing the input length under the *contiguous* setting may be beneficial for accuracy, while doing so under the *disconnected* setting would not, as the temporal contiguity is broken. However, if GPI is a relevant factor, they may benefit from longer input sequences under both settings, as the disconnected context may still be informative about the series' generative process. We carry out this experiment on 4 real-world datasets: LargeST-SD [30] (*traffic*), CEREn [31] (*energy*), ClmUS-T and ClmUS-R (*climate*, see App. G). Fig. 2 reports the MAE for increasing values of W under the contiguous setting (orange bars), and compares it to increasing values of C under the disconnected setting (blue bars). For the latter, we set $S = 336$: a separation of two weeks between context and window.

Reasonably, the disconnected context should not carry relevant temporal dependencies in the sense of CF, while still being representative of the same process that generated the window (assuming the process does not abruptly change). Across most datasets, the two settings yield analogous error trends (example forecasts in App. K.1), suggesting that both models may use part of the input to implicitly infer the target data-generating process. Note that performance gaps between the two settings are expected, given that these models are trained on temporally contiguous inputs and that concatenating disconnected segments may introduce artifacts at their junction.

3.2 Process identification and window length

The analysis in Sec. 3.1 shows that, in global models, increasing the window length W can lead to more accurate forecasts by reducing uncertainty about which specific process is generating the n-th target time series. Indeed, Eq. 5 shows the role of the input window when forecasting over a group of time series generated by different processes. Optimal predictions can be seen as a weighted average across likely processes; longer observation windows allow for tailoring predictions to the target time series by contracting the posterior $p\left(\boldsymbol{\phi}^n \mid \cdot\right)$. Note that this effect does not exist in local models trained on data coming from a single time series.

To illustrate the impact of GPI on global forecasting tasks, we consider as a minimal example the case in which the target processes are simple Markovian processes of order P, with $P \in \mathbb{N}^+$. This will provide theoretical grounding for the discussion we will later extend to real-world scenarios. In particular, we consider the sub-family of processes in Eq. 3 such that the family $\mathcal{G}$ is determined by

$$x_t^n = g\big(\boldsymbol{x}_{t-P:t}^n; \boldsymbol{\phi}^n\big) + \epsilon_t^n, \quad \text{where} \quad \boldsymbol{\phi}^n \sim \boldsymbol{\mathcal{P}}_{\boldsymbol{\phi}}, \text{ and } \epsilon_t^n \sim \boldsymbol{\mathcal{P}}_{\epsilon}, \quad \forall t. \tag{6}$$

Here, $g\left(\,\cdot\,; \boldsymbol{\phi}^n\right)$ is a well-behaved parametric function, such that each resulting process is stationary. Noise terms ϵ_t^n are i.i.d. , sampled from time-invariant distribution $\boldsymbol{\mathcal{P}}_{\epsilon}$ with finite variance and zero mean. In this setting, x_t^n clearly depends only on the most recent P observations, and a window of length P would capture all relevant temporal dependencies.

Assumption 3.1. *Assume that data are generated as in Eq. 6 such that*

$$\mathbb{E}\big[\mathrm{Var}_{\boldsymbol{\phi}^n}\big(\mathbb{E}\big[x_t^n \mid \boldsymbol{x}_{t-P:t}^n, \boldsymbol{\phi}^n\big] \mid \boldsymbol{x}_{t-P:t}^n\big)\big] > 0.$$

Assumption 3.1 states that the expected value of the process variance given the last P observations is greater than zero. Said differently, we assume that processes corresponding to different parameters $\boldsymbol{\phi}^n$ cannot be uniquely identified from the most recent P observations and that, given such observations, there are plausible processes with different conditional expectations. This is a very reasonable assumption, especially when training on related (noisy) time series. It holds whenever the same sequence of P observations can be generated by different processes, but the conditional expectation for the next observation differs across them.

Proposition 3.2. [Necessity of $W > P$.] *Under the formulation in Eq. 6, and Assumption 3.1, let $F\left(\boldsymbol{x}_{t-W:t}^n; \boldsymbol{\Theta}_{opt}\right)$ denote an optimal 1-step-ahead predictor (in the sense of Eq. 4) for input window length W.*

*Then, a window size $W > P$ is a **necessary** condition for $F\left(\boldsymbol{x}_{t-W:t}^n; \boldsymbol{\Theta}_{opt}\right)$ to achieve the minimum expected error* $\mathrm{Var}\left(\epsilon_t^n\right)$.

The proof of Proposition 3.2 is given in App. D.1. The takeaway is that, regardless of how far back the true temporal dependencies go (i.e., how large P is), GPI introduces an overhead, as we might need more than P observations to attain the minimum possible expected error. Indeed, as shown in App. D.1, the optimal expected error of a predictor with window length $W \geq P$ can be written as

$$\mathbb{E}\left[\mathrm{Var}_{\phi^n}\left(\mathbb{E}\left[x_t^n \middle| \boldsymbol{x}_{t-P:t}^n, \boldsymbol{\phi}^n\right] \middle| \boldsymbol{x}_{t-W:t}^n\right)\right] + \mathrm{Var}\left(\epsilon_t^n\right). \tag{7}$$

Here, the right term is the irreducible error for any window length W, i.e., the variance of the noise. The left term, instead, reflects the expected error due to the variance in the conditional expectation of x_t^n across parameter values $\boldsymbol{\phi}^n$ that could likely have generated $\boldsymbol{x}_{t-W:t}^n$. Notably, beyond accounting for temporal dependencies, using longer observation windows can enable more accurate predictions by reducing the first term in Eq. 7, provided additional observations contract the posterior $p\left(\boldsymbol{\phi}^n \mid \cdot\right)$. This suggests that a key motivation for increasing window length in global models is to improve GPI, rather than strictly to capture long-range dependencies. In fact, in practice, achieving accurate

forecasts may require an input window size W which substantially exceeds the relevant temporal dependencies of the target data-generating process. This is driven by the uncertainty about $\boldsymbol{\phi}^n$ implied by $\mathcal{P}_{\boldsymbol{\phi}}$ and by how rapidly this uncertainty shrinks as W grows. However, alongside higher computational costs, increasing the input size often introduces additional challenges, e.g., increases model complexity. This trade-off is ever more relevant when considering time series foundation models, which are trained on heterogeneous data coming from a large variety of domains.

Empirical analysis The need for a window length exceeding the temporal order of the target process can also be investigated empirically through simulations in controlled environments. We generate a dataset of $N = 1000$ time series, each sampled from a different *nonlinear autoregressive* (NAR) process of order $P = 2$ (see App. G.1). We train both global and local models (implemented as a single-layer *gated recurrent unit* (GRU) [32]) for input windows of varying length W. The global model's parameters are shared across all time series, while the local model is trained from scratch on each time series. Fig. 3 reports the MSE achieved in the transductive (left) and inductive (right) settings (see App. G.2 for details on dataset splits). Note that local models are inherently transductive-only. As one would expect, the local model (green curve) achieves near-optimal error (red line) already at $W = P = 2$, while the global model (orange curve) approaches it only for $W \gg P$. As each generating process has order $P = 2$ by construction, we can explain the additional observations needed by the global model as the effect of uncertainty about the target data-generating process.

In the next step, we train a variant of the global GRU model (details in App. H.1.1) which, similar to the experiment in Fig. 2, takes as input a sequence obtained by concatenating a past *context* $\boldsymbol{x}^n_{t'-C:t'}$ of length C and a recent *window* $\boldsymbol{x}^n_{t-W:t}$ of length W. During training the context's location t' is drawn uniformly from $[t-W-S, t-W)$ to avoid bias towards specific temporal lags, while at test time it is fixed to $t-W-S$. From the analysis in Sec. 3.1, we expect that GPI can be addressed by leveraging any sequence generated by the target process, not only those adjacent to the target x^n_t. In fact, under the formulation in Eq. 6, $p\big(\boldsymbol{\phi}^n \,\big|\, \boldsymbol{x}^n_{t-W:t}\big)$ is invariant to time shifts. The blue curve in Fig. 3 corresponds to the results of this experiment, using $S = 50$ as the maximum temporal gap between window and context. Notably, adding the context enables near-minimum error already at $W = P$, supporting the idea that, even if here additional observations beyond the last P time lags are not relevant for their temporal relationship with x^n_t, they indeed serve to reduce uncertainty about the target data-generating process (additional results in App. K.2, including experiments for $P > 2$).

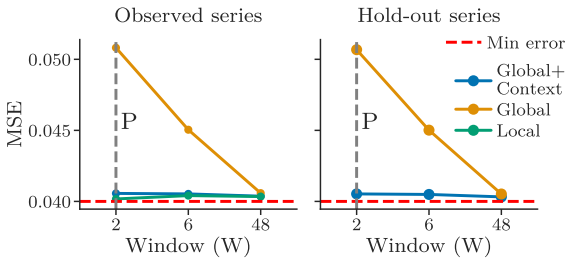


Figure 3: 1-step-ahead forecasting error (MSE, 3 runs, $\pm std$) for NAR processes. $P = 2$, $C = 46$.

3.3 The cost of GPI in foundation models

Consider the typical use case for a foundation model: it is pretrained over multiple domains, then typically used *zero-shot* by different users on a specific target domain. Although this flexibility is clearly an advantage, according to our analysis in Sec. 3.2, it might entail a cost in terms of the required input window length, compared to a domain-specific global model. Indeed, we expect the target data distribution for a single end user to have much narrower support and lower variance than a foundation model's pretraining distribution. However, in the absence of a mechanism to efficiently condition the foundation model on the desired target distribution, the input sequence length required to resolve uncertainty about the target process will still depend on the pretraining distribution. Indeed, by looking at published results, foundation models [15, 29] require longer observation windows than global models trained directly on the target domain [11, 33, 34, 35]. The following experiments highlight this phenomenon associated with global models.

Empirical analysis: controlled environments To investigate how training on multiple domains impacts global models, we design the following controlled experiment. We simulate different domains by making each domain correspond to NAR processes of a specific order $P \in \{1, 2, 4, 6\}$. For each domain, we generate a dataset of $N = 500$ time series, where the parameters of the process generating

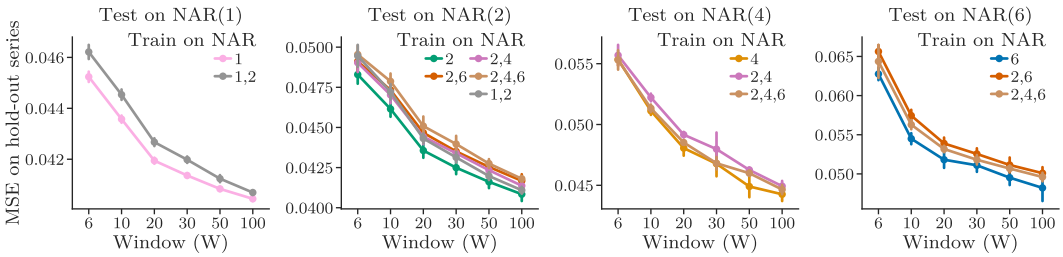


Figure 4: 1-step-ahead forecasting (MSE, *inductive*, 3 runs $\pm std$) for different combinations of NAR domains. Columns show error on a specific domain, color identifies the training domains.

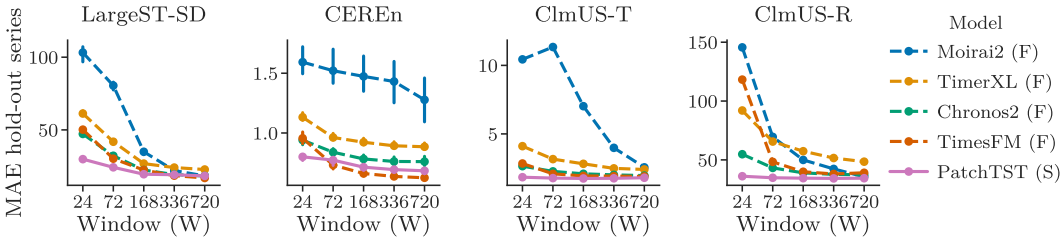


Figure 5: 24-step-ahead forecasting error (MAE, *inductive*, 3 runs $\pm std$) against window length W. Foundation models (F, *dashed*) against domain-specific global models (S, *solid*).

each series are randomly sampled from a uniform distribution. We then train a single-layer GRU, from scratch, either on a single dataset (single-domain) or jointly on multiple datasets (multi-domain).[1] Finally, each model is separately evaluated on a hold-out set of unseen time series for each of the domains they were trained on. Results are summarized in Fig. 4. Each subplot corresponds to the test performance on a specific domain (i.e., processes of a specific order). Colors denote different training data distributions (either single-domain or multi-domain). Points on each curve correspond to varying the input window length hyperparameter W (additional results in App. K.3). The results show that, when considering performance on a specific domain for a fixed W, training on multiple domains consistently leads to worse performance w.r.t. training directly on the target domain. We attribute this to increased uncertainty about the specific process generating the observed input sequence when the training data include multiple domains. At the same time, increasing W allows models trained jointly on multiple domains to match, or surpass, the performance of models trained on a single domain. We attribute this effect to a reduction of the uncertainty about the target data-generating process given longer input sequences.

Empirical analysis: real-world We extend the analysis to real-world scenarios by considering the datasets from Sec. 3.1 and comparing foundation models to domain-specific global models in *inductive* settings. We consider Moirai2, TimerXL, Chronos2 [23], and TimesFM2.5 [14] as foundation models, and PatchTST [11] as a representative architecture for the domain-specific models (details in App. H.1.2). For each dataset and varying window lengths W, we train PatchTST from scratch (details in App. H.2) using only time series from the target dataset. Finally, for each dataset and window length W, we compare PatchTST's performance to that of the foundation models on time series entirely held out during training. Note that the foundation models' pretraining corpora contain time series from the considered domains. Fig. 5 reports the experiment's results. Subplots denote the dataset, line color denotes the model, and line style denotes whether the model is foundation (dashed) or domain-specific (solid). Clearly, pretraining on a very heterogeneous data distribution results in the foundation models requiring a much larger input window to match the performance of the domain-specific models (additional baselines in App. K.3). We now extend the previous experiment to show that this effect is not specific to models trained on vast multi-domain datasets (foundation models). For each window length W we additionally train a PatchTST model in a multi-domain fashion on the union of the four considered real-world datasets, using the same hyperparameters as the domain-specific PatchTST. We evaluate the multi-domain PatchTST's error at varying window lengths W on each dataset separately, considering again the *inductive* setting where only entirely

[1] Both single- and multi-domain training involve multiple data-generating processes, so both settings would involve GPI.

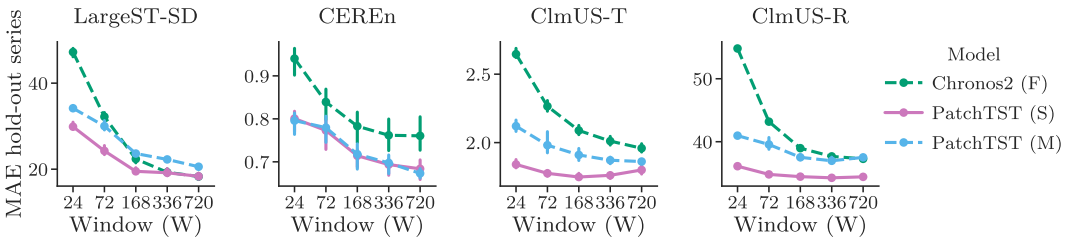


Figure 6: 24-step-ahead forecasting error (MAE, *inductive*, 3 runs $\pm std$) against window length W. Foundation model (F), against multi-domain model (M), against domain-specific global model (S).

held-out series are used for testing. Fig. 6 reports the results of the experiment, comparing the multi-domain PatchTST to the domain-specific ones. We also include the error curve of Chronos2 as a reference for foundation models. Overall, the number of training domains appears linked to the trade-off between input length and forecasting error: models trained across all four datasets (M) yield worse trade-offs than domain-specific global models (S), but better ones than a foundation model (F).

These results confirm that the flexibility of foundation models comes at a cost, a cost clearly linked to GPI. Additional results in App. F confirm that GPI might require higher model capacity than CF. In resource-constrained environments, considering these aspects is relevant and can make a domain-specific alternative preferable to a foundation model.

4 Leveraging insights for model design

We now show how our analysis can inform model design and provide practical advantages in real-world settings. For this purpose, we demonstrate a methodology to amortize the inference cost associated to GPI. Recall insights from Sec. 3.1: (i) GPI can require larger observation windows than CF; (ii) GPI may rely on sequences generated by the target process, but not adjacent to the target x_t^n. Therefore, we can try to improve model scalability by separating GPI and CF.

Model design We aim at (partially) decoupling CF and GPI by using predictors of the form:

$$\boldsymbol{e}^n = G\big(\boldsymbol{x}^n_{t'-C:t'};\, \boldsymbol{\Theta}_G\big) \tag{8a}$$

$$\hat{\boldsymbol{x}}^n_{t:t+H} = F\big(\boldsymbol{x}^n_{t-W:t},\, \boldsymbol{e}^n;\, \boldsymbol{\Theta}_F\big) \tag{8b}$$

where $t' < t - W$ and $C > W$. Here, $G(\cdot)$ takes as input a *context* $\boldsymbol{x}^n_{t'-C,t'}$ drawn from the target time series history, and outputs a latent representation $\boldsymbol{e}^n$; $F(\cdot)$, instead, operates on the *window* $\boldsymbol{x}^n_{t-W,t}$ containing the latest observations, and it is further conditioned on $\boldsymbol{e}^n$. The rationale is that $G(\cdot)$ can produce an embedding $\boldsymbol{e}^n$, summarizing information about the process generating the n-th time series. This can be cached and re-used for subsequent forecasts of the same time series, reducing the inference cost compared to the standard contiguous window approach using $C+W$ observations. In fact, at inference time, predicting a target time series for m steps would require processing $C+m \cdot W$ observations, instead of $m \cdot (C+W)$. In applications where GPI can require much more information than CF, e.g., foundation models, we can set $C \gg W$ to increase the number of observations for GPI, with no impact on the cost of repeated forecasts for the same time series. Similarly, if one expects GPI to require more capacity than CF, the architecture of $G(\cdot)$ can be more complex than that of $F(\cdot)$. Note that, if the data-generating process drifts over time, one could regularly recompute $\boldsymbol{e}^n$ from a more recent context.

Models of the form in Eq. 8 can be advantageous w.r.t. popular architectures for global models. For non-recurrent models (e.g., *multilayer perceptron*s, *encoder-only* Transformers) the computational advantage is clear, as they need to process the entire input sequence for each forecast. Most recurrent global models, instead, are based on input-patching (e.g., *decoder-only* Transformers), where the input window is split into multiple patches of length L. In this case, even if the architecture is recurrent, for a new forecast, one either needs to wait and collect L new observations, or the entire input sequence must be processed from scratch. This can increase the inference cost for end users of foundation models, as these models are mostly patch-based Transformers [17], but the patch-size L cannot be tuned to the needs of any specific application. The formulation in Eq. 8 can help manage

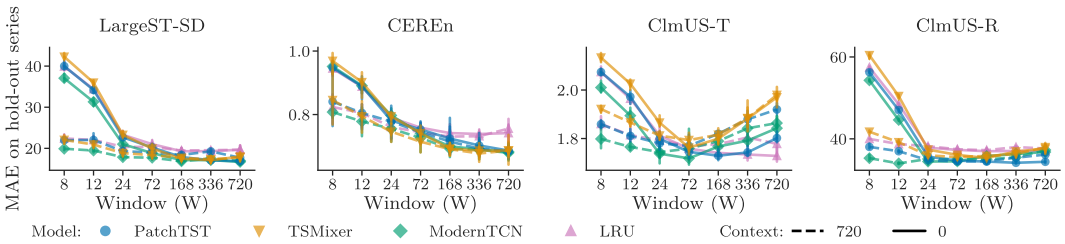


Figure 7: 24-step-ahead forecasting error (MAE, *inductive*, 3 runs $\pm std$). Color/marker denotes the model. Line-style denotes the approach: decoupling (dashed) or standard (solid).

cost-to-performance trade-offs arising from GPI, regardless of the implementation of $G(\cdot)$ and $F(\cdot)$ (further discussion in App. I). Finally, this decoupled design naturally accommodates intermittent data streams, where long contiguous windows of observations might not be available.

Empirical analysis We quantify the performance-to-cost trade-offs through evaluation on the real-world datasets introduced in Sec. 3.1. We implement the *base model* in Eq. 8b with representative state-of-the-art architectures: PatchTST [11] (attention), TSMixer [34] (fully connected), ModernTCN [35] (convolutional), and *linear recurrent unit* (LRU) [36] (recurrent); details in App. H.1.2. The *embedding module* (Eq. 8a) is always implemented by PatchTST for consistency. Depending on the implementation of the base model, $\boldsymbol{e}^n$ is integrated via summation or concatenation (see App. H.1.2). For each base model, hyperparameters are the same for both the standard and decoupled versions (details in App. H.2). When training decoupled models (Eq. 8), $\boldsymbol{x}^n_{t'-C:t'}$ is drawn uniformly from $[t-W-S, t-W)$, where S specifies a maximum back-shift to prevent sampling outdated contexts. Randomizing the temporal distance between t' and t also prevents bias toward specific time lags. During testing, we draw the context using the maximum back-shift considered during training.

Fig. 7 reports the MAE obtained when training each model to forecast the next $H=24$ time-steps. Points on the curve denote results from training with different window lengths W, using context length $C=720$ and maximum back-shift $S=336$. Line color denotes the base model. Solid lines denote the performance of the standard state-of-the-art (SOTA) architectures (implementing Eq. 1), while dashed lines are their counterparts with input decoupling (Eq. 8). Across the considered datasets and models, the MAE of the decoupled approach at small values of W is often comparable to that achieved by the base model at larger W values. For higher W values, the MAE for the two strategies is similar. In such a case, $\boldsymbol{x}^n_{t-W:t}$ might be already sufficient to specialize the model's predictions.

In Fig. 8 we report performance-cost Pareto curves comparing the standard approach (orange curve) and the decoupled approach (blue curve). We consider models from the previous experiment, and plot their MSE against inference speed, *floating-point operations* (FLOPs), and memory cost, considering the case where $\boldsymbol{e}^n$ is kept fixed across multiple forecasts. Each point in the plots corresponds to a specific architecture, dataset, window size, and implementation design (either standard or decoupled). The MSE is normalized per dataset, while cost metrics are normalized per-architecture (see App. K.4 for detailed per model/dataset pair results). We can observe how the decoupled approach is the Pareto optimal one. Overall, these results support the practical applicability of our insights, suggesting that model scalability can be improved by considering GPI and CF during model design.

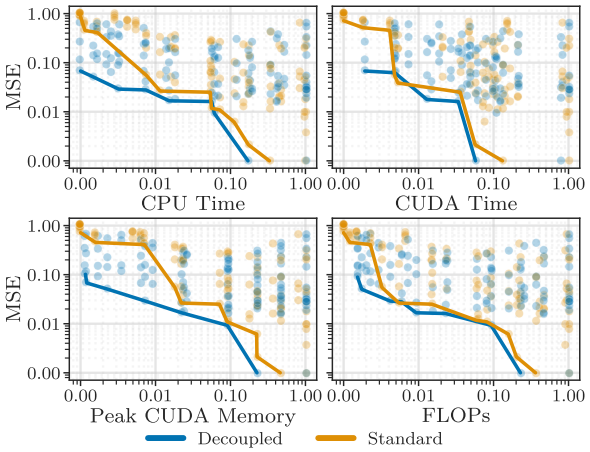


Figure 8: Pareto frontiers for the decoupled (blue) and standard (orange) approach. Dots correspond to model/dataset pairs from Fig. 7 (3-run average). MSE (y-axis, *inductive*) is min-max normalized per dataset, while cost metrics (x-axis) per architecture.

Discussion Ultimately, our results advocate for a change in the design perspective for time series foundation models. Instead of relying on large contiguous input windows and monolithic architectures, one could design architectures decoupling GPI and CF. For example, GPI could benefit from metadata about the target time series. Lightweight downstream predictors could be tailored via reusable embeddings extracted from rich contexts. Considering GPI and CF distinctly can broaden the design space for time series models towards more flexible and efficient designs.

5 Related Works

The implications of training a single model on large collections of time series, rather than having models specific to each time series, have often been overlooked. Montero-Manso and Hyndman [9] note how transductive global models need longer input windows to match the performance of local approaches. We expand their findings, leveraging GPI to provide an explanation of the mechanisms by which increasing the window size enables better performance also in inductive scenarios (e.g., foundation models [37]). Recent works started exploring ICL [22] for time series forecasting, with approaches mostly relying on exogenous information. Faw et al. [38] use related time series signals, Auer et al. [39], Ansari et al. [23] adopt covariate signals, while Williams et al. [40] feed the model textual descriptors. Notably, Lu et al. [41] instead augment the model input with example past/future pairs from the target time series itself. In our framework, the effectiveness of these approaches can be explained with the role played by GPI.

Finally, in transductive learning settings, different works on *global-local* models [42, 43, 44] show how global models can be augmented with local components to efficiently address GPI at training time. In this work, we address inductive settings where one cannot afford local components, and discuss how purely global models can instead leverage the input sequence to address GPI at inference time. See App. J for further discussion of related works.

6 Conclusion

In this paper, we analyze key aspects of deep learning for time series forecasting, showing that effective prediction hinges on the interplay between GPI and CF. Our theoretical and empirical analyses clarify that long observation windows can improve performance by improving GPI, and show that model design choices encouraging the decoupling of the mentioned tasks can improve scalability. Altogether, our insights can inform the next generation of time series foundation models.

Limitations and Future works While we demonstrate the practical relevance of our insights, we leave the development and evaluation of a foundation model based on said insights as future work. In this regard, future research should consider more sophisticated approaches, e.g., based on dynamically selecting the number of recent observations to process to balance accuracy and efficiency in CF. Moreover, investigating methodologies to decouple GPI in the context of highly time-varying data-generating processes is another interesting direction for future studies. Proposition 3.2 does not quantify the overhead due to GPI in practical scenarios. Future work might consider exploring a more detailed theoretical characterization of how the overhead depends on the characteristics of the target data distribution.

Acknowledgments and Disclosure of Funding

GDF acknowledges support from the Swiss National Science Foundation (SNSF) through the grant 205121_197242 for the project "PROSELF: Semi-automated Self-Tracking Systems to Improve Personal Productivity", and from the Hasler Foundation through the project "Towards Scalable Multimodal Causal Deep Learning" (grant ID 2024-05-15-70). AC was supported by the SNSF with project no. 225351, "Relational Deep Learning for Reliable Time Series Forecasting at Scale".

References

[1] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems*, 34:22419–22430, 2021.

[2] Konstantinos Benidis, Syama Sundar Rangapuram, Valentin Flunkert, Yuyang Wang, Danielle Maddix, Caner Turkmen, Jan Gasthaus, Michael Bohlke-Schneider, David Salinas, Lorenzo Stella, et al. Deep learning for time series forecasting: Tutorial and literature survey. *ACM Computing Surveys*, 55(6):1–36, 2022.

[3] Yuxuan Wang, Haixu Wu, Jiaxiang Dong, Yong Liu, Mingsheng Long, and Jianmin Wang. Deep time series models: A comprehensive survey and benchmark. *arXiv preprint arXiv:2407.13278*, 2024.

[4] George EP Box and Gwilym M Jenkins. Some recent advances in forecasting and control. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 17(2):91–109, 1968.

[5] Jan G. De Gooijer and Rob J. Hyndman. 25 years of time series forecasting. *International Journal of Forecasting*, 22(3):443–473, 2006. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2006.01.001. URL https://www.sciencedirect.com/science/article/pii/S0169207006000021. Twenty five years of forecasting.

[6] 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.

[7] 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.

[8] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The m4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1):54–74, 2020. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2019.04.014. URL https://www.sciencedirect.com/science/article/pii/S0169207019301128. M4 Competition.

[9] 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.

[10] 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 *The Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Virtual Conference*, volume 35, pages 11106–11115. AAAI Press, 2021.

[11] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *The Eleventh International Conference on Learning Representations*, 2022.

[12] 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.

[13] 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*, 2022. URL https://openreview.net/forum?id=0EXmFzUn5I.

[14] Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In *Forty-first International Conference on Machine Learning*, 2024.

[15] Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. In *International Conference on Machine Learning*, pages 53140–53164. PMLR, 2024.

[16] Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. Moment: A family of open time-series foundation models. *arXiv preprint arXiv:2402.03885*, 2024.

[17] Yuxuan Liang, Haomin Wen, Yuqi Nie, Yushan Jiang, Ming Jin, Dongjin Song, Shirui Pan, and Qingsong Wen. Foundation models for time series analysis: A tutorial and survey. In *Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining*, pages 6555–6565, 2024.

[18] Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit bayesian inference. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=RdJVFCHjUMI.

[19] Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. What learning algorithm is in-context learning? investigations with linear models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=0g0X4H8yN4I.

[20] Yingcong Li, Muhammed Emrullah Ildiz, Dimitris Papailiopoulos, and Samet Oymak. Transformers as algorithms: Generalization and stability in in-context learning. In *International conference on machine learning*, pages 19565–19594. PMLR, 2023.

[21] Yufeng Zhang, Fengzhuo Zhang, Zhuoran Yang, and Zhaoran Wang. What and how does in-context learning learn? bayesian model averaging, parameterization, and generalization. In Yingzhen Li, Stephan Mandt, Shipra Agrawal, and Emtiyaz Khan, editors, *Proceedings of The 28th International Conference on Artificial Intelligence and Statistics*, volume 258 of *Proceedings of Machine Learning Research*, pages 1684–1692. PMLR, 03–05 May 2025. URL https://proceedings.mlr.press/v258/zhang25d.html.

[22] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui. A survey on in-context learning. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1107–1128, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.64. URL https://aclanthology.org/2024.emnlp-main.64/.

[23] Abdul Fatir Ansari, Oleksandr Shchur, Jaris Küken, Andreas Auer, Boran Han, Pedro Mercado, Syama Sundar Rangapuram, Huibin Shen, Lorenzo Stella, Xiyuan Zhang, et al. Chronos-2: From univariate to universal forecasting. *arXiv preprint arXiv:2510.15821*, 2025.

[24] Tilmann Gneiting. Making and evaluating point forecasts. *Journal of the American Statistical Association*, 106(494):746–762, 2011.

[25] Andrew Gelman, John B Carlin, Hal S Stern, and Donald B Rubin. *Bayesian data analysis*. Chapman and Hall/CRC, 1995.

[26] Rishabh Agarwal, Avi Singh, Lei M Zhang, Bernd Bohnet, Luis Rosias, Stephanie C.Y. Chan, Biao Zhang, Aleksandra Faust, and Hugo Larochelle. Many-shot in-context learning. In *ICML 2024 Workshop on In-Context Learning*, 2024. URL https://openreview.net/forum?id=goi7DFHlqS.

[27] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[28] Chenghao Liu, Taha Aksu, Juncheng Liu, Xu Liu, Hanshu Yan, Quang Pham, Doyen Sahoo, Caiming Xiong, Silvio Savarese, and Junnan Li. Moirai 2.0: When less is more for time series forecasting. *arXiv preprint arXiv:2511.11698*, 2025.

[29] Yong Liu, Guo Qin, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Timer-xl: Long-context transformers for unified time series forecasting. In *The Thirteenth International Conference on Learning Representations*, 2025.

[30] Xu Liu, Yutong Xia, Yuxuan Liang, Junfeng Hu, Yiwei Wang, Lei Bai, Chao Huang, Zhenguang Liu, Bryan Hooi, and Roger Zimmermann. Largest: A benchmark dataset for large-scale traffic forecasting. *Advances in Neural Information Processing Systems*, 36:75354–75371, 2023.

[31] Commission for Energy Regulation. CER. CER Smart Metering Project - Electricity Customer Behaviour Trial, 2009-2010 [dataset]. *Irish Social Science Data Archive. SN: 0012-00*, 2016. URL `https://www.ucd.ie/issda/data/commissionforenergyregulationcer/`.

[32] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–decoder approaches. In Dekai Wu, Marine Carpuat, Xavier Carreras, and Eva Maria Vecchi, editors, *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/W14-4012. URL `https://aclanthology.org/W14-4012/`.

[33] 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.

[34] Si-An Chen, Chun-Liang Li, Sercan O Arik, Nathanael Christian Yoder, and Tomas Pfister. Tsmixer: An all-mlp architecture for time series forecast-ing. *Transactions on Machine Learning Research*, 2023.

[35] Donghao Luo and Xue Wang. Moderntcn: A modern pure convolution structure for general time series analysis. In *The twelfth international conference on learning representations*, pages 1–43, 2024.

[36] 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.

[37] Siva Rama Krishna Kottapalli, Karthik Hubli, Sandeep Chandrashekhara, Garima Jain, Sunayana Hubli, Gayathri Botla, and Ramesh Doddaiah. Foundation models for time series: A survey. *arXiv preprint arXiv:2504.04011*, 2025.

[38] Matthew Faw, Rajat Sen, Yichen Zhou, and Abhimanyu Das. In-context fine-tuning for time-series foundation models. In *Forty-second International Conference on Machine Learning*, 2025.

[39] Andreas Auer, Raghul Parthipan, Pedro Mercado, Abdul Fatir Ansari, Lorenzo Stella, Bernie Wang, Michael Bohlke-Schneider, and Syama Sundar Rangapuram. Zero-shot time series forecasting with covariates via in-context learning. *arXiv preprint arXiv:2506.03128*, 2025.

[40] Andrew Robert Williams, Arjun Ashok, Étienne Marcotte, Valentina Zantedeschi, Jithendaraa Subramanian, Roland Riachi, James Requeima, Alexandre Lacoste, Irina Rish, Nicolas Chapados, and Alexandre Drouin. Context is key: A benchmark for forecasting with essential textual information. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu, editors, *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 66887–66944. PMLR, 13–19 Jul 2025. URL `https://proceedings.mlr.press/v267/williams25a.html`.

[41] Jiecheng Lu, Yan Sun, and Shihao Yang. In-context time series predictor. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=dCcY2pyNIO`.

[42] 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.

[43] 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.

[44] Luca Butera, Giovanni De Felice, Andrea Cini, and Cesare Alippi. On the regularization of learnable embeddings for time series forecasting. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=F5ALCh3GWG.

[45] Zhe Li, Shiyi Qi, Yiduo Li, and Zenglin Xu. Revisiting long-term time series forecasting: An investigation on linear mapping. *arXiv preprint arXiv:2305.10721*, 2023.

[46] Giovanni De Felice, Andrea Cini, Daniele Zambon, Vladimir V Gusev, and Cesare Alippi. Graph-based virtual sensing from sparse and partial multivariate observations. In *International Conference on Learning Representations*, 2024.

[47] Taha Aksu, Gerald Woo, Juncheng Liu, Xu Liu, Chenghao Liu, Silvio Savarese, Caiming Xiong, and Doyen Sahoo. Gift-eval: A benchmark for general time series forecasting model evaluation. *arXiv preprint arXiv:2410.10393*, 2024.

[48] Yong Liu, Haoran Zhang, Chenyu Li, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Timer: generative pre-trained transformers are large time series models. In *Proceedings of the 41st International Conference on Machine Learning*, pages 32369–32399, 2024.

[49] Abdul Fatir Ansari, Lorenzo Stella, Ali Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, et al. Chronos: Learning the language of time series. *Transactions on Machine Learning Research*, 2024.

[50] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[51] 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.

[52] Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1ecqn4YwB.

[53] Tian Zhou, Peisong Niu, Xue Wang, Liang Sun, and Rong Jin. One fits all: Power general time series analysis by pretrained LM. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=gMS6FVZvmF.

[54] Kashif Rasul, Arjun Ashok, Andrew Robert Williams, Arian Khorasani, George Adamopoulos, Rishika Bhagwatkar, Marin Biloš, Hena Ghonia, Nadhir Hassen, Anderson Schneider, et al. Lag-llama: Towards foundation models for time series forecasting. In *R0-FoMo: Robustness of Few-shot and Zero-shot Learning in Large Foundation Models*, 2023.

[55] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[56] Duo Chai, Wei Wu, Qinghong Han, Fei Wu, and Jiwei Li. Description based text classification with reinforcement learning. In *International conference on machine learning*, pages 1371–1382. PMLR, 2020.

[57] Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*, 2022.

[58] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[59] Yanda Chen, Ruiqi Zhong, Sheng Zha, George Karypis, and He He. Meta-learning via language model in-context tuning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 719–730, 2022.

[60] Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. Metaicl: Learning to learn in context. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2791–2809, 2022.

[61] Weijia Shi, Sewon Min, Maria Lomeli, Chunting Zhou, Margaret Li, Xi Victoria Lin, Noah A Smith, Luke Zettlemoyer, Wen-tau Yih, and Mike Lewis. In-context pretraining: Language modeling beyond document boundaries. In *The Twelfth International Conference on Learning Representations*, 2024.

[62] Pablo Montero-Manso, George Athanasopoulos, Rob J Hyndman, and Thiyanga S Talagala. Fforma: Feature-based forecast model averaging. *International Journal of Forecasting*, 36(1): 86–92, 2020.

[63] Jürgen Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, 1992.

[64] David Ha, Andrew M Dai, and Quoc V Le. Hypernetworks. In *International Conference on Learning Representations*, 2017.

[65] Boris N Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. Meta-learning framework with applications to zero-shot time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 9242–9250, 2021.

[66] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR, 06–11 Aug 2017. URL `https://proceedings.mlr.press/v70/finn17a.html`.

[67] Thiyanga S Talagala, Rob J Hyndman, and George Athanasopoulos. Meta-learning how to forecast time series. *Journal of Forecasting*, 42(6):1476–1501, 2023.

[68] Declan A Norton, Edward Ott, Andrew Pomerance, Brian Hunt, and Michelle Girvan. Tailored forecasting from short time series via meta-learning. *arXiv preprint arXiv:2501.16325*, 2025.

A Reproducibility statement

Code to reproduce the experiments in the paper is openly available on GitHub at `https://github.com/LucaButera/wlcw`. Regarding the datasets used in the paper, LargeST-SD [30] is openly available, CEREn [31] requires a data access request to the appropriate authority, while ClmUS-T and ClmUS-R will be provided alongside the code. The experiments on foundation models require getting access to the pre-trained weights for Moirai2 [28], TimerXL [29], Chronos2 [23], and TimesFM2.5 [14]. These are openly available online. The code is complemented by a README which explains how to set up the Python environment and reproduce the experiments. Required assets are automatically downloaded when needed. Moreover, for reproducibility purposes, randomized operations in the code are controlled by fixed seeds.

B Broader impact statement

Our insights have the potential to guide the design of the next generation of foundation models for time series forecasting towards more principled approaches with better scalability. The decomposition into two sub-problems, GPI and CF, could also improve interoperability with related tasks, e.g., imputation, anomaly detection. In this sense, this work can significantly impact the field of deep learning for time series and result in positive downstream societal impacts. However, we believe that our work does not add any ethical or societal concern for harmful impacts w.r.t. those that apply to global models for time series at large.

C Terminology

We distinguish here several related terms.

- ***Generative process identification*** refers to inferring which latent stochastic process generated an observed sequence, without necessarily recovering parameters.
- **System identification** usually denotes estimating explicit parametric models of dynamical systems, often with control in mind.
- **Dynamics identification** emphasizes learning the functional form governing state transitions.
- **Model identification** refers more broadly to choosing or estimating a model structure consistent with observed data.
- **Process characterization** targets statistical properties of the series (e.g., stationarity, ergodicity, dependence structure) rather than the underlying dynamics.
- **In-context learning** denotes the ability of a deep learning model to adapt to a new task on the fly, without parameter updates, by conditioning on examples of the target task.

Throughout this work, we use *generative process identification* (GPI) in this narrow sense of reducing the uncertainty about the underlying generating process given a finite context window.

D Proofs

D.1 Proof of Proposition 3.2

[Necessity of $W > P$.] Under the formulation in Eq. 6, and Assumption 3.1, let $F\left(\boldsymbol{x}_{t-W:t}^{n}; \boldsymbol{\Theta}_{opt}\right)$ denote an optimal 1-step-ahead predictor (in the sense of Eq. 4) for input window length W.

Then, a window size $W > P$ is a **necessary** condition for $F\left(\boldsymbol{x}_{t-W:t}^{n}; \boldsymbol{\Theta}_{opt}\right)$ to achieve the minimum expected error $\mathrm{Var}\left(\epsilon_t^n\right)$.

Proof. We proceed by contradiction. Assume that there exists a predictor $F\left(\cdot; \boldsymbol{\Theta}_{opt}\right)$ that achieves the minimum possible expected error on every process in $\mathcal{G}$ (Eq. 6). We will show that under the $W \leq P$ assumption, such predictor does not exist.

We consider the two cases:

Case $W < P$. This case is immediate. Even for a fixed process, an order-P Markov model may require the last P observations to achieve the minimum possible expected error. Without additional assumptions guaranteeing that the conditional mean is a function of only W lags, no predictor using only $\boldsymbol{x}^n_{t-W:t}$ can, in general, recover the conditional expectation.

Case $W = P$**.** Consider any predictor $F(\boldsymbol{x}^n_{t-P:t}; \boldsymbol{\Theta})$ that observes only the last P values. Its expected MSE under the data-generating mechanism induced by $\boldsymbol{\phi}^n \sim \boldsymbol{\mathcal{P}}_{\boldsymbol{\phi}}$ and Eq. 6 is

$$\mathcal{R}(\boldsymbol{\Theta}) \;\coloneqq\; \mathbb{E}\Big[\big(x^n_t - F(\boldsymbol{x}^n_{t-P:t}; \boldsymbol{\Theta})\big)^2\Big]. \tag{9}$$

By Eq. 4, an MSE-optimal predictor satisfies

$$F(\boldsymbol{x}^n_{t-P:t}; \boldsymbol{\Theta}_{opt}) \simeq \mathbb{E}\big[x^n_t \mid \boldsymbol{x}^n_{t-P:t}\big]. \tag{10}$$

Hence, the minimum attainable expected error among predictors is

$$\mathcal{R}(\boldsymbol{\Theta}_{opt}) = \mathbb{E}\Big[\big(x^n_t - \mathbb{E}\big[x^n_t \mid \boldsymbol{x}^n_{t-P:t}\big]\big)^2\Big] \tag{11}$$

$$= \mathbb{E}_{\boldsymbol{x}^n_{t-P:t}}\big[\mathrm{Var}\big(x^n_t \mid \boldsymbol{x}^n_{t-P:t}\big)\big]. \tag{12}$$

We now expand the conditional variance by conditioning on the latent parameter $\boldsymbol{\phi}^n$. By the law of total variance,

$$\mathrm{Var}\big(x^n_t \mid \boldsymbol{x}^n_{t-P:t}\big) = \\ \mathbb{E}_{\boldsymbol{\phi}^n}\big[\mathrm{Var}_{\epsilon^n_t}\big(x^n_t \mid \boldsymbol{x}^n_{t-P:t}, \boldsymbol{\phi}^n\big) \;\big|\; \boldsymbol{x}^n_{t-P:t}\big] \;+\; \mathrm{Var}_{\boldsymbol{\phi}^n}\big(\mathbb{E}_{\epsilon^n_t}\big[x^n_t \mid \boldsymbol{x}^n_{t-P:t}, \boldsymbol{\phi}^n\big] \;\big|\; \boldsymbol{x}^n_{t-P:t}\big). \tag{13}$$

Under the signal-plus-noise model in Eq. 6, with ϵ^n_t independent of $\boldsymbol{\phi}^n$ and $\boldsymbol{x}^n_{<t}$ (hence of $\boldsymbol{x}^n_{t-P:t}$), and with time-invariant variance, the first term reduces to

$$\mathrm{Var}_{\epsilon^n_t}\big(x^n_t \mid \boldsymbol{x}^n_{t-P:t}, \boldsymbol{\phi}^n\big) = \mathrm{Var}_{\epsilon^n_t}\big(\epsilon^n_t \mid \boldsymbol{x}^n_{t-P:t}, \boldsymbol{\phi}^n\big) = \mathrm{Var}(\epsilon^n_t)$$

Substituting into Eq. 13 yields

$$\mathrm{Var}\big(x^n_t \mid \boldsymbol{x}^n_{t-P:t}\big) = \mathrm{Var}(\epsilon^n_t) + \mathrm{Var}_{\boldsymbol{\phi}^n}\big(\mathbb{E}_{\epsilon^n_t}\big[x^n_t \mid \boldsymbol{x}^n_{t-P:t}, \boldsymbol{\phi}^n\big] \mid \boldsymbol{x}^n_{t-P:t}\big). \tag{14}$$

Taking $\mathbb{E}_{\boldsymbol{x}^n_{t-P:t}}[\cdot]$, this gives

$$\mathcal{R}(\boldsymbol{\Theta}_{opt}) = \mathrm{Var}(\epsilon^n_t) + \mathbb{E}_{\boldsymbol{x}^n_{t-P:t}}\big[\mathrm{Var}_{\boldsymbol{\phi}^n}\big(\mathbb{E}_{\epsilon^n_t}\big[x^n_t \mid \boldsymbol{x}^n_{t-P:t}, \boldsymbol{\phi}^n\big] \mid \boldsymbol{x}^n_{t-P:t}\big)\big]. \tag{15}$$

However, by the proposition's assumptions

$$\mathbb{E}_{\boldsymbol{x}^n_{t-P:t}}\big[\mathrm{Var}_{\boldsymbol{\phi}^n}\big(\mathbb{E}_{\epsilon^n_t}\big[x^n_t \mid \boldsymbol{x}^n_{t-P:t}, \boldsymbol{\phi}^n\big] \mid \boldsymbol{x}^n_{t-P:t}\big)\big]) > 0.$$

Therefore,

$$\mathcal{R}(\boldsymbol{\Theta}_{opt}) > \mathrm{Var}_{\epsilon^n_t}(\epsilon^n_t),$$

so no predictor with $W = P$ can attain the minimum expected error $\mathrm{Var}_{\epsilon^n_t}(\epsilon^n_t)$ for every target process.

□

E Example with AR(2)

Example E.1 (two AR(2) processes)**.** *Consider the case where the system model in Sec. 2 is fully specified and defined as*

$$\begin{cases} x^n_t = \phi^n_1 \cdot x^n_{t-1} + \phi^n_2 \cdot x^n_{t-2} + \epsilon^n_t \\ \epsilon^n_t \sim \mathcal{N}\left(0, \sigma^2\right), \quad \boldsymbol{\phi}^n \sim \mathit{Categorical}\left((\boldsymbol{\phi}^{n_1}, 0.5), (\boldsymbol{\phi}^{n_2}, 0.5)\right) \end{cases}, \forall t, \tag{16}$$

where $\boldsymbol{\phi}^n = (\phi^n_1, \phi^n_2)$. In this case, processes in $\mathcal{G}$ are autoregressive (AR) of order $P = 2$; the noise follows a zero-mean Gaussian distribution with the same variance for all possible processes, and $\boldsymbol{\mathcal{P}}_{\boldsymbol{\phi}}$ is a categorical distribution assigning uniform probability to two possible parameter values $\boldsymbol{\phi}^{n_1}$ and $\boldsymbol{\phi}^{n_2}$, which we assume to correspond to stationary AR processes.

$\mathbf{W} > \mathbf{P}$ is necessary to achieve the minimum expected error for every target process: *Let $F\left(x_{t-1}^n, x_{t-2}^n; \boldsymbol{\Theta}_{opt}\right)$ denote an optimal predictor (in the sense of Eq. 4) for input window length $W = 2$, trained via MSE minimization.*

Then, for every input window $\left(x_{t-1}^n, x_{t-2}^n\right)$, the predictor approximates the conditional expectation $\mathbb{E}_{\epsilon_t^n}\left[x_t^n \,\middle|\, x_{t-1}^n, x_{t-2}^n\right]$. By the law of total expectation, this conditional expectation can be extended by further conditioning on the latent categorical random variable $\boldsymbol{\phi}^n$.

$$F\left(x_{t-1}^n, x_{t-2}^n; \boldsymbol{\Theta}_{opt}\right) \simeq \mathbb{E}_{\epsilon_t^n}\left[x_t^n \,\middle|\, x_{t-1}^n, x_{t-2}^n\right] \tag{17}$$

$$= \mathbb{E}_{\boldsymbol{\phi}^n}\left[\mathbb{E}_{\epsilon_t^n}\left[x_t^n \,\middle|\, x_{t-1}^n, x_{t-2}^n, \boldsymbol{\phi}^n\right] \,\middle|\, x_{t-1}^n, x_{t-2}^n\right] \tag{18}$$

Let us define:

$$\pi := p\left(\boldsymbol{\phi}^n = \boldsymbol{\phi}^{n_1} \,\middle|\, x_{t-1}^n, x_{t-2}^n\right) \tag{19}$$

as the posterior probability of the process parameters taking value $\boldsymbol{\phi}^{n_1}$. The outer expectation can be expanded as follows:

$$F\left(x_{t-1}^n, x_{t-2}^n; \boldsymbol{\Theta}_{opt}\right) \simeq \pi \cdot \mathbb{E}_{\epsilon_t^n}\left[x_t^n \,\middle|\, x_{t-1}^n, x_{t-2}^n, \boldsymbol{\phi}^n = \boldsymbol{\phi}^{n_1}\right]$$
$$+ (1-\pi) \cdot \mathbb{E}_{\epsilon_t^n}\left[x_t^n \,\middle|\, x_{t-1}^n, x_{t-2}^n, \boldsymbol{\phi}^n = \boldsymbol{\phi}^{n_2}\right] \tag{20}$$

$$= \pi \cdot \left(\phi_1^{n_1} x_{t-1}^n + \phi_2^{n_1} x_{t-2}^n\right) + (1-\pi) \cdot \left(\phi_1^{n_2} x_{t-1}^n + \phi_2^{n_2} x_{t-2}^n\right) \tag{21}$$

$$= \underbrace{\left(\pi\phi_1^{n_1} + (1-\pi)\,\phi_1^{n_2}\right)}_{\bar{\phi}_1} x_{t-1}^n + \underbrace{\left(\pi\phi_2^{n_1} + (1-\pi)\,\phi_2^{n_2}\right)}_{\bar{\phi}_2} x_{t-2}^n \tag{22}$$

Eq. 22 shows that an optimal predictor behaves as a convex combination of the two possible AR parameter values, where the weights of the combination are dynamically adapted to the input window depending on π. Considering Eq. 15 (see App. D.1), for any target process following Eq. 16, the optimal predictor can attain the minimum possible expected error $\mathrm{Var}_{\epsilon_t^n}\left(\epsilon_t^n\right)$ if and only if

$$\mathbb{E}_{x_{t-1}^n, x_{t-2}^n}\left[\mathrm{Var}_{\boldsymbol{\phi}^n}\left(\mathbb{E}_{\epsilon_t^n}\left[x_t^n \,\middle|\, x_{t-1}^n, x_{t-2}^n, \boldsymbol{\phi}^n\right] \mid x_{t-1}^n, x_{t-2}^n\right)\right] = 0$$

This condition is only satisfied if, for all possible $\left(x_{t-1}^n, x_{t-2}^n\right)$, one of the following edge cases is met:

- $\pi = 1$ *and* $\boldsymbol{\phi}^n = \boldsymbol{\phi}^{n_1}$ *(or* $\pi = 0$ *and* $\boldsymbol{\phi}^n = \boldsymbol{\phi}^{n_2}$*)*
- $\phi_1^{n_1} \cdot x_{t-1}^n + \phi_2^{n_1} \cdot x_{t-2}^n = \phi_1^{n_2} \cdot x_{t-1}^n + \phi_2^{n_2} \cdot x_{t-2}^n$

In the first case, input observations unequivocally determine the parameters that generated them, while, in the second case, the two possible values $\boldsymbol{\phi}^n$ can take imply the same conditional expectation for x_t^n. Notably, this latter case includes the trivial scenario where $\boldsymbol{\phi}^{n_1} = \boldsymbol{\phi}^{n_2}$ ($\boldsymbol{\mathcal{P}}_{\boldsymbol{\phi}}$ is degenerate). Otherwise, under input length $W = P = 2$, an optimal predictor cannot attain the minimum expected error. In fact, there would exist cases where

$$\boldsymbol{\phi}^n = \boldsymbol{\phi}^{n_1} \quad \textit{and} \quad F\left(x_{t-1}^n, x_{t-2}^n; \boldsymbol{\Theta}_{opt}\right) \neq \phi_1^{n_1} x_{t-1}^n + \phi_2^{n_1} x_{t-2}^n \quad \textit{or}$$
$$\boldsymbol{\phi}^n = \boldsymbol{\phi}^{n_2} \quad \textit{and} \quad F\left(x_{t-1}^n, x_{t-2}^n; \boldsymbol{\Theta}_{opt}\right) \neq \phi_1^{n_2} x_{t-1}^n + \phi_2^{n_2} x_{t-2}^n.$$

To show the alignment between the theory and the empirical case we generate a synthetic dataset by randomly selecting $\boldsymbol{\phi}^{n_1}$ and $\boldsymbol{\phi}^{n_2}$, and train a neural network to minimize the MSE for the 1-step-ahead prediction. Specifically, we use a network of the following form:

$$[p^{n_1}, p^{n_2}] = softmax\left(\boldsymbol{W}_2 \sigma\left(\boldsymbol{W}_1 \boldsymbol{x}_{t-2:t} + b_1\right) + b_2\right), \tag{23}$$

$$\hat{x}_t^n = \left(p^{n_1}\psi_1^{n_1} + p^{n_2}\psi_1^{n_2}\right) x_{t-1}^n + \left(p^{n_1}\psi_2^{n_1} + p^{n_2}\psi_2^{n_2}\right) x_{t-2}^n, \tag{24}$$

with learnable parameters $\boldsymbol{W}_1 \in \mathbb{R}^{4\times 2}$, $\boldsymbol{W}_2 \in \mathbb{R}^{2\times 4}$, b_1, b_2, $\psi_1^{n_1}$, $\psi_2^{n_1}$, $\psi_1^{n_2}$ and $\psi_2^{n_2} \in \mathbb{R}^1$. Here σ denotes the Rectified Linear Unit activation function.

Fig. 9 compares predictions obtained with different methods on a hold-out set of test inputs $\left(x_{t-1}^n, x_{t-2}^n\right)$. The rightmost plot compares predictions produced by the learned neural network with those obtained by computing Eq. 22 in closed form, showing a clear alignment. The first three plots from the left compare the ground truth x_t^n to the prediction of three different strategies: neural network (third from the left), closed form Eq. 22 (second from the left) and computing

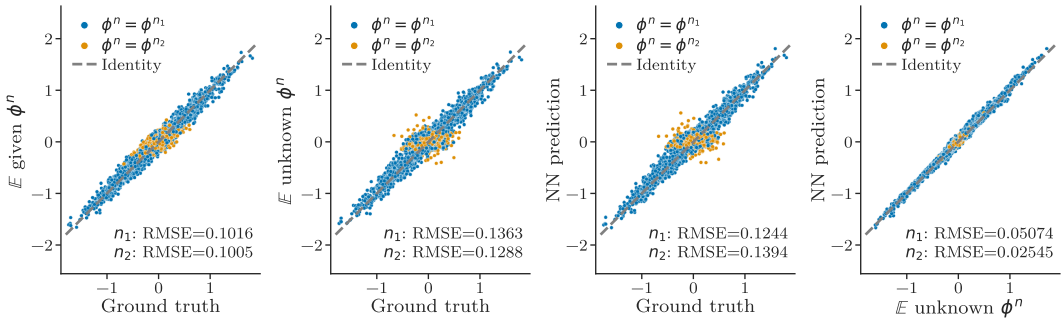


Figure 9: Parity plots for Example E.1. Data generation hyper-parameters: $\sigma^2 = 0.1$, $seed = 7$, $\#timesteps = 10000$. *Ground truth* denotes the actual realizations from the stochastic process. $\mathbb{E}$ *given* $\boldsymbol{\phi}^n$ denotes the expected value for the future observation when knowing the process parameters and the last 2 observations. $\mathbb{E}$ *unknown* $\boldsymbol{\phi}^n$ denotes the predictions obtained by using the real process parameters weighted by the likelihood in Eq. 19 computed in closed form. *NN prediction* denotes the predictions obtained by a neural network optimized to minimize the MSE. The *root mean squared error* (RMSE) is computed between the two variables on each plot's axes, and reported separately for processes with $\boldsymbol{\phi}^n = \boldsymbol{\phi}^{n_1}$ and $\boldsymbol{\phi}^n = \boldsymbol{\phi}^{n_2}$.

$\mathbb{E}_{\epsilon_t^n}\left[x_t^n \,\middle|\, x_{t-1}^n, x_{t-2}^n, \boldsymbol{\phi}^n\right]$ *given knowledge about the value of* $\boldsymbol{\phi}^n$ *for the process that generated the input (first from the left). We can see how the first two strategies are worse (higher spread around the identity line) than computing the expected value given knowledge about* $\boldsymbol{\phi}^n$*. App. K.5 shows additional results for different synthetic dataset seeds. These results highlight how, in this example case: (i) a neural model can behave as Eq. 22; (ii) an input window of length* $W = P = 2$ *is not sufficient to achieve the minimum attainable error on both possible processes at once.*

F Process identification and model capacity

In this section we discuss how addressing GPI can require models with more capacity than that which would be required by CF if the "identity" of the target process was known in advance, e.g., a predictor for a single time series.

Linear processes We start by considering the simple scenario of stationary linear AR processes of order $P = 2$ (which can be generalized to $P \in \mathbb{N}^+$)

$$x_t^n = \phi_1^n \cdot x_{t-1}^n + \phi_2^n \cdot x_{t-2}^n + \epsilon_t^n, \quad \epsilon_t^n \sim \mathcal{N}\left(0, \sigma^2\right). \tag{25}$$

In this case process parameters $\phi^n = (\phi_1^n, \phi_2^n)$ follow the target distribution $\boldsymbol{\mathcal{P}}_{\boldsymbol{\phi}}$. For simplicity, consider the case of a predictor with input window length $W \geq P$. The optimal output for such predictor corresponds to $\mathbb{E}\left[x_t^n \,\middle|\, \boldsymbol{x}_{t-W:t}^n\right]$. According to Eq. 5 we expect the optimal prediction $\mathbb{E}\left[x_t^n \,\middle|\, \boldsymbol{x}_{t-W:t}^n\right]$ to be a non-linear function of $\boldsymbol{x}_{t-W:t}^n$, even if the individual target processes are linear. Consider, for instance, the example in App. E, where the conditional expectation involves the product of two functions of $\boldsymbol{x}_{t-W:t}^n$ (Eq. 20). However, if the target distribution $\boldsymbol{\mathcal{P}}_{\boldsymbol{\phi}}$ implies only a single possible process, $\mathbb{E}\left[x_t^n \,\middle|\, \boldsymbol{x}_{t-W:t}^n\right]$ would be linear in $\boldsymbol{x}_{t-W:t}^n$. As such we expect a linear model to be not expressive enough to approximate $\mathbb{E}\left[x_t^n \,\middle|\, \boldsymbol{x}_{t-W:t}^n\right]$ if multiple distinct linear AR processes are possible under $\boldsymbol{\mathcal{P}}_{\boldsymbol{\phi}}$.

To show this effect empirically, we train[2] both global and local linear models over a group of linear AR(2) processes generated according to App. G.1, considering different amounts of training processes and different input window lengths. The results of the experiment are reported in Fig. 10. As one would expect, the local approach reaches near-minimum error with $W = 2$ (left subplot). However, we can observe that the global one is unable to close this gap, even when the input sequence is very long ($W = 300$). Increasing the number of processes in the training dataset improves inductive performance (right subplot), likely by providing a richer sample of the target distribution. However, overall the error remains higher w.r.t. the local approach. In line with our previous discussion,

[2]We learn the linear models by using *ordinary least squares*.

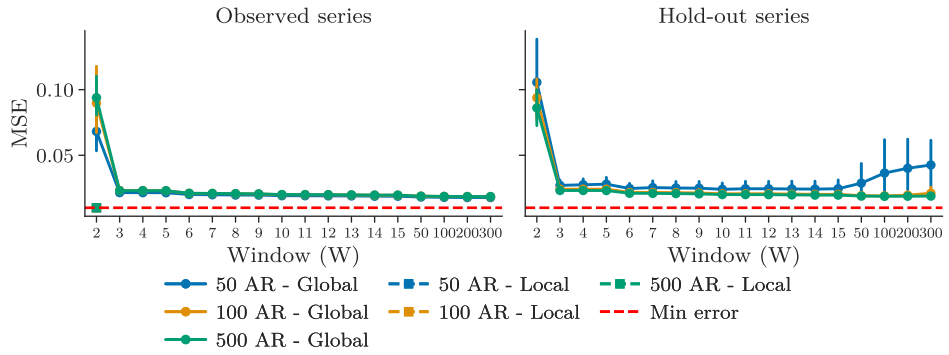


Figure 10: Performance of a linear model on data from multiple AR(2) processes with $\sigma^2 = 0.1$ (MSE, 5 runs $\pm std$). For each process we generate 10000 observations and use an $80\% - 20\%$ train-test split across the temporal axis. Held-out data come from a fixed set of 200 AR(2).

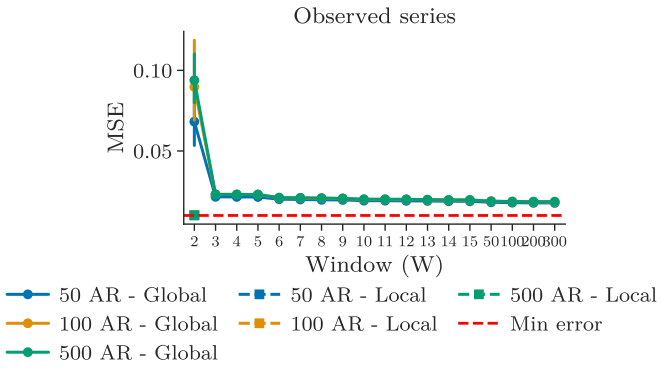
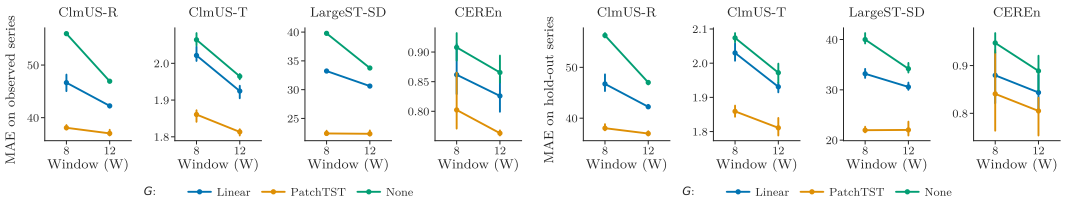


Figure 11: Performance of a linear model on data from multiple AR(2) processes with $\sigma^2 = 0.1$ (MSE, 5 runs $\pm std$). For each process we generate 10000 observations. Global models use an $80\% - 20\%$ train-test split across the temporal axis. Local models are trained using only a subsequence of 300 observations from each time series and tested on the same data as global models.

these results suggest that, even if the target family of generative processes is linear, a linear model lacks the expressivity to address GPI. To further substantiate this, we train local models using just a subsequence of 300 observations from each time series, corresponding the maximum input window length used for global models. Fig. 11 shows an equivalent of the left subplot in Fig. 10, where local models have been learned on this reduced data amount and tested on the same data as global counterparts. Notably, local models achieve near-minimum error also in this scenario.

Real-world data To expand the analysis on how addressing GPI can require higher model capacity than CF, we now consider real-world settings. Linear models have been shown to perform competitively w.r.t. complex models in transductive settings [33], but may struggle on benchmarks where the number of training time series is large [45]. A possible interpretation for this phenomenon is that, in many real-world tasks, a linear model may be suitable for one or few fixed time series, however they may struggle when GPI is more complex, e.g., inductive scenarios. To asses this we consider the decoupled modeling approach of Sec. 4 and carry out an experiment similar to that in Fig. 7. Here, however, we use a linear model, i.e., a linear layer, to implement $G(\cdot)$ (Eq. 8a), and we compare it against implementing $G(\cdot)$ by means of PatchTST (a transformer-based neural network), or using the standard modeling approach without $G(\cdot)$ (Eq. 1). Fig. 12 reports the results of the experiment across different datasets, when considering PatchTST as the implementation of $F(\cdot)$ (Eq. 8b). We can observe that, while using a linear layer for $G(\cdot)$ improves performance over the standard approach, it yields consistently higher error compared to using PatchTST for $G(\cdot)$. This suggests that the complexity of GPI could be a leading factor in determining the capacity needed by global forecasting models even in real-world settings.



(a) Transductive setting.

(b) Inductive setting.

Figure 12: 24-step-ahead forecasting error (MAE, 3 runs $\pm std$) for different implementations of $G(\cdot)$ (Eq. 8a) in the *decoupled* approach of Sec. 4. *None* refers to the *standard* approach. $F(\cdot)$ (Eq. 8b) is always PatchTST.

G Datasets

Table 1: Details of the considered real-world dataset.

Dataset	# of time series	# of time steps	# of channels	Sampling rate
CEREn	960	12,865	1	1 hour
LargeST-SD	716	52,128	1	1 hour
ClmUS-T	1000	26,304	1	1 hour
ClmUS-R	1000	26,304	1	1 hour
Electricity	321	26,304	1	1 hour
Traffic	862	17,544	1	1 hour
Weather	1	52,696	21	10 minutes

CEREn dataset of electric load readings from [31]. Contains energy consumption readings from smart meters across small and medium enterprises in Ireland. Data was collected during the Smart Metering Project, supervised by the Commission for Energy Regulation (CER). Access to the data has to be requested through `https://www.ucd.ie/issda/data/commissionforenergyregulationcer/`. We considered a subset of the available smart meters, those for small and medium enterprises of type 3 (SME_CODE) with at most 1% of missing readings. Data spans ~ 1.5 years.

LargeST-SD large scale traffic dataset from Liu et al. [30]. Contains traffic flow readings from loop detectors across the California state highway system. We consider the *SD* sub-dataset, which contains only sensors in the San Diego area. Licensed under Attribution 4.0 International (CC BY 4.0). Data span ~ 6 years.

ClmUS-T and ClmUS-R dataset of climate variables from satellite readings, inspired by that from De Felice et al. [46]. Contains data from the POWER Project's Daily 2.3.5 version on 2023/02/26 and sampled from a grid of 1000 coordinates spanning the entire surface of the United States. Visit the project's website (`https://power.larc.nasa.gov/`) for further information and programmatic access to the data. In particular, ClmUS-T consists of hourly *mean temperature* ($^\circ C$) readings, while ClmUS-R considers the *allsky surface shortwave irradiance* (W/m^2). Data spans 3 years.

Electricity dataset of hourly electricity consumption readings from Wu et al. [1]. The dataset consists of univariate time series from 321 customers, spanning the years from 2012 to 2014. The dataset is released under the MIT license.

Traffic dataset of hourly road occupancy rates from Wu et al. [1]. Data is collected from the California Department of Transportation, and consists of measurements at 862 locations across the San Francisco Bay area freeways. The data span a period of 2 years, from 2016 to 2018. The dataset is released under the MIT license.

Weather dataset of meteorological indicators (e.g., humidity, temperature) from Wu et al. [1]. Data are recorded every 10 minutes over the entire year of 2020, and consist of a single multivariate time series with 21 channels. The dataset is released under the MIT license.

G.1 Synthetic datasets

To build synthetic datasets, we generate time series from NAR processes of order P using the following generative model:

$$\begin{cases} x_t^i = \tanh\left(\sum_{p=1}^{P} \phi_p^i x_{t-p}^i\right) + \epsilon_t^i \\ \left[\phi_1^i, \dots, \phi_P^i\right] \sim \mathcal{U}\left(-\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right) \\ \epsilon_t^i \sim \mathcal{N}(0, 0.2) \end{cases} \quad , \forall\, i, t, \tag{26}$$

where $\mathcal{U}\left(a, b\right)$ denotes a P-dimensional uniform distribution over the interval $[a, b]$, while $\mathcal{N}\left(\mu, \sigma^2\right)$ denotes a gaussian distribution with mean μ and variance σ^2.

When considering time series generated by *linear* AR processes of order $P = 2$, we drop the $\tanh$ from Eq. 26. Moreover, we sample process parameters as $\left[\phi_1^i, \phi_2^i\right] = \left[\cos\left(\alpha^i\right), \sin\left(\alpha^i\right)\right]$, with $\alpha^i \sim \mathcal{U}\left(0, 2\pi\right)$, and keep only samples where $\left|\phi_1^i - \phi_2^i\right| < 1$ and $\left(\phi_1^i + \phi_2^i\right) < 1$. The purpose of this is to ensure the system's stability, i.e., avoid collapse to $\pm\infty$, for the linear case, where there is no $\tanh$ to keep the output bounded.

Synthetic data scale For the experiment in Fig. 3 we generate 10000 time-steps for each process, across a total of 1000 processes. Conversely, for the experiment in Fig. 4, which involved combining multiple datasets, we generate 5000 time-steps per process, across a total of 500 processes per dataset.

G.2 Data splitting

Table 2: Data splitting strategy.

	# Timesteps		
# Series	Training Data	In-sample Validation	In-sample Test
		Out-of-sample Validation	
			Out-of-sample Test

It is important to note that we split datasets for *train*, *validation* and *test* both along the temporal axis and the 'series' axis. Meaning that we first select some time series which will be entirely held-out for *inductive*, i.e., out-of-sample, validation and testing. Then we take the series in the training set, and we hold-out the latest realizations for *transductive*, i.e., in-sample, validation and testing. Table 3, reports the actual number of time series and time steps, per series, we used for each split in real-world datasets. Note that, for synthetic datasets we used a 70/10/20 train/val/test split both for the temporal and 'series' axis. Table 2 visually explains the adopted splitting strategy. To reduce bias due to data splitting, experiments repeated across multiple runs have random splits along the 'series' axis controlled by the run's seed. Note that, in synthetically generated datasets, the run's seed also controls the randomized operations in data generation (e.g., data-generating process parameters, noise sampling).

Note that we use both the in-sample and out-of-sample validation sets for early-stopping, in order to avoid over-fitting the in-sample series. We do this for all experiments where we train a model, by taking the mean loss between the two sets, in order to avoid assigning more importance to a set over the other due to difference in number of series.

H Experimental details

This section provides detailed information about model implementation, training and used hyper-parameters.

Table 3: Data splitting details for the considered real-world datasets. IS stands for *in-sample*, OOS stands for *out-of-sample*.

	Number of time series			Number of time-steps		
	Train/IS Val/IS Test	OOS Val	OOS Test	Train	IS/OOS Val	IS/OOS Test
CEREn	672	96	192	8760	2052	2053
LargeST-SD	572	72	72	6132	876	1752
ClmUS-T	700	100	200	17520	4392	4392
ClmUS-R	700	100	200	17520	4392	4392
Electricity	225	32	64	18414	2630	5260
Traffic	604	86	172	12282	1754	3508
Weather	21	N/A	N/A	36888	5269	10539

H.1 Model implementation

This section provides additional details regarding the implementation of the models we adopted in our experiments.

H.1.1 GRU and GRU with context

In our experiment we used the standard GRU [32] architecture, described as

$$\begin{aligned}
\boldsymbol{r}_t^n &= \sigma\left(\boldsymbol{W}_r^x x_t^n + \boldsymbol{b}_r^x + \boldsymbol{W}_r^h \boldsymbol{h}_{t-1}^n + \boldsymbol{b}_r^h\right) \\
\boldsymbol{z}_t^n &= \sigma\left(\boldsymbol{W}_z^x x_t^n + \boldsymbol{b}_z^x + \boldsymbol{W}_z^h \boldsymbol{h}_{t-1}^n + \boldsymbol{b}_z^h\right) \\
\boldsymbol{m}_t^n &= \tanh\left(\boldsymbol{W}_m^x x_t^n + \boldsymbol{b}_m^x + \boldsymbol{r}_t^n \odot \left(\boldsymbol{W}_m^h \boldsymbol{h}_{t-1}^n + \boldsymbol{b}_m^h\right)\right) \\
\boldsymbol{h}_t^n &= (1 - \boldsymbol{z}_t^n) \odot \boldsymbol{m}_t^n + \boldsymbol{z}_t^n \odot \boldsymbol{h}_{t-1}^n,
\end{aligned}$$

where $\boldsymbol{W}$'s and $\boldsymbol{b}$'s denote the learnable weights of linear layers and σ denotes the sigmoid activation function. In particular, note that $x_t^n \in \mathbb{R}$ is the input signal for the n-th time series at time t, while $\boldsymbol{h}_t^n \in \mathbb{R}^{d_h}$ is the hidden state for the n-th time series at time t. Thus we use the following shorthand notation to denote the recursive application of a GRU, of learnable parameters $\boldsymbol{\Psi}$, over a sequence $\boldsymbol{x}_{t-W:t}^n$, with initial state $\boldsymbol{h}_0$,

$$\boldsymbol{h}_t^n = GRU\left(\boldsymbol{x}_{t-W:t}^n, \boldsymbol{h}_0; \boldsymbol{\Psi}\right).$$

Unless differently specified, $\boldsymbol{h}_0$ is initialized to a vector of zeros.

In the experiments where only the contiguous *window* over $[t - W, t)$ is present, our model can be described as

$$\begin{aligned}
\boldsymbol{h}_t^n &= GRU\left(\boldsymbol{x}_{t-W:t}^n, \boldsymbol{h}_0; \boldsymbol{\Psi}\right), \\
\hat{\boldsymbol{x}}_{t:t+H}^n &= MLP\left(\boldsymbol{h}_t^n; \boldsymbol{\Phi}\right),
\end{aligned}$$

where $MLP\left(\cdot; \boldsymbol{\Phi}\right) : \mathbb{R}^{d_h} \to \mathbb{R}^H$ denotes a simple *multilayer perceptron* (MLP), with one hidden layer and hidden size d_h, mapping the last hidden state to the prediction horizon length. Conversely, when we consider also the *context* sequence, spanning $[t' - C, t')$, with $t' < t - W$, we use a model of the following form

$$\begin{aligned}
\boldsymbol{h}_{t'}^n &= GRU\left(\boldsymbol{x}_{t'-C:t'}^n, \boldsymbol{h}_0; \boldsymbol{\Psi}_C\right), \\
\boldsymbol{h}_t^n &= GRU\left(\boldsymbol{x}_{t-W:t}^n, \boldsymbol{h}_{t'}^n; \boldsymbol{\Psi}_W\right), \\
\hat{\boldsymbol{x}}_{t:t+H}^n &= MLP\left(\boldsymbol{h}_t^n; \boldsymbol{\Phi}\right).
\end{aligned}$$

This implementation simply entails learning dedicated weights to process the *window* or the *context* and using the last state $\boldsymbol{h}_{t'}^n$, obtained by processing the *context*, as initial state for the GRU which will process the *window*. The motivation behind this implementation is informing the model whether it is processing the latest observations preceding the target or not. Nonetheless, this can also be achieved by other means, e.g., using a standard GRU and inserting a separator value between context and window.

Table 4: Hyper-parameters used for the SOTA reference architectures adopted in the paper.

Model	Hidden size (d_h)	# of layers	Feedforward size	# of heads	Patch size
PatchTST	16	3	128	4	16
ModernTCN	64	4	512	-	-
LRU	64	6	-	-	-
TSMixer	64	2	-	-	-

H.1.2 State-of-the-art architectures

All the SOTA architectures we consider in this work follow the original paper implementation, thus, for each one we report relevant hyper-parameters (Tab. 4) alongside an explanation of how $\boldsymbol{e}^n \in \mathbb{R}^{d_e}$ (Eq. 8) was given as input to $F(\cdot)$. Notice that in all our experiments we fix $d_e = 64$.

PatchTST PatchTST [11] is a standard *encoder-only* transformer, which, given a sequence of length W, transforms it into patches of length W_p, which are then projected, through a linear layer, to size d_h. As such, we simply map $\boldsymbol{e}^n \in \mathbb{R}^{d_e}$ to size d_e, by means of a linear layer, and then sum its value to each input patch representation.

ModernTCN ModernTCN [35] is a convolution based architecture, where the first $1D$ convolution layer maps the input signal $\boldsymbol{x}^n_{t-W:t}$ from size d_x to size d_h. Right after this initial layer, we sum $\boldsymbol{e}^n$ to the latent representations, broadcasting along the temporal dimension. If $d_e \neq d_h$, we map it to the appropriate dimension by means of a linear layer.

LRU LRU [36] is a recurrent architecture, so, in the same fashion as we do for the GRU (H.1.1), we use $\boldsymbol{e}^n$ as initial state for the recurrence. In this case, since LRU operates on complex numbers, we first map $\boldsymbol{e}^n$ to a complex number in $\mathbb{C}^{d_h}$ by means of two linear layers, one for the real and one for the imaginary part.

TSMixer TSMixer [34] is the only architecture that already considered, in its design, a computation path to provide *static attributes* as input to the model. Given this, we simply considered $\boldsymbol{e}^n$ as a static attribute. We refer to the original paper for an explanation of how such attributes are integrated in the computation.

H.1.3 Foundation models

Table 5: Hyper-parameters of the time series foundation models considered in the paper.

Model	Hidden size	# of layers	Feedforward size	# of heads	Patch size
Moirai2	384	6	1024	6	16
TimerXL	1024	8	4096	8	96
Chronos2	512	6	2048	8	16
TimesFM2.5	1280	20	1280	16	32

For the foundation models considered in the paper, we consider the original implementation and pre-trained weights. We report the relevant hyper-parameters in Tab. 5. Note that, when using sequence lengths that are not a multiple of the models' patch size, we adopt the padding or masking strategy adopted by the original implementations. Following is a brief description of the architectures and their pretraining data.

Moirai2 Moirai2 [28] is a decoder-only Transformer with causal attention. It is pretrained on the GiftEVAL Pretrain dataset [47], a large-scale dataset spanning time series from several real-world domains for a total of ≈ 230 billion data-points (i.e., individual time-steps).

TimerXL TimerXL [29] uses a decoder-only Transformer architecture with causal attention. It is pretrained on both the LOTSA dataset [15] and the UTSD dataset [48] for a total of ≈ 260 billion data-points. Both datasets span time series from several real-world application domains.

Chronos2 Chronos2 [23] uses an encoder-decoder Transformer architecture with full attention. It is pretrained on a subset of datasets from the Chronos training data [49] and the GiftEVAL Pretrain dataset [47], plus additional synthetically generated time series. In our experiments we consider the *small* variant of the model.

TimesFM2.5 TimesFM2.5 [14] is a decoder-only Transformer with causal attention. It is pretrained on a large scale dataset consisting of over 100 billion time-points from real-world time series, plus additional synthetically generated data. In our experiments we consider the 200 million parameters variant of the model.

H.2 Training details

All models are trained using the MSE as loss function and the *Adam* optimizer [50]. When *context* is used, the maximum back-shift allowed, during training, is $S = 336$, which corresponds to context at most 2 weeks old for the considered data with hourly frequency. During testing the back-shift is fixed to 336 for consistency. The only exception is the experiment reported in Fig. 3, where this number is set to 50, however the choice of this parameter is not critical in such scenario.

We use sinusoidal temporal encodings with daily period for the experiments in Sec. 4, these encodings are commonly used to inject a notion of temporal distance among data-points in deep learning models for time series. We train all models for 50 epochs, with early-stopping, with patience of 8 epochs and scheduled learning rate reduction with a factor of 0.1 after 5 epochs without loss improvement.

We use a batch size of 2560, and limit the number of train batches based on dataset size. In particular, we use 1140 for CEREn, 2880 for LargeST-SD and 1200 for ClmUS-T and ClmUS-R. For experiments with GRUs on synthetic data we use a learning rate of 0.01. For models trained on real-world data we fix the learning rate to 0.001, except for *ModernTCN*-based architectures, for which we set the learning rate to 0.00025 on CEREn, ClmUS-T, and ClmUS-R to avoid optimization instability.

Computing resources We ran our experiments on A5000 and Titan V NVIDIA GPUs. All of our experiments fit within 24 GBs of VRAM, but a large part of them can even run on more moderate hardware, with 12 GBs of VRAM. By reducing the batch size one could even stay within the very contained amount of < 8 GBs of VRAM. A single experiment, i.e., training a single architecture on a single dataset, can take up to 4 hours, considering the largest datasets and more demanding models. However, some experiments, like those in the controlled environments, take few minutes to be executed. In the context of a single A5000 NVIDIA GPU, with 24 GBs of VRAM, we estimate the total amount of compute-hours to carry out the experiments in the paper, to be ≈ 1800h (≈ 75 days).

I Fixed context and caching

Setting the context length One of the advantages of the decoupled approach in Sec. 4 is that one can set different values for C (i.e., the *context* length) and W (i.e., the *window* length), allocating a different sequence length to the model components intended for GPI and CF respectively. In our experiments in Sec. 4, C is a hyperparameter which is kept fixed in order to study how additional context affects model performance as W varies. In practical applications, we expect one to search for a suitable value of C as it is usually done for hyperparameters, e.g., by grid- or random-search. While we do not explore the question of how to efficiently search for suitable values of C and W jointly, here we sketch a possible heuristic. One could set C to a large value (depending on the application and available sample size) and reduce W until significant performance drops are observed. Then fix W to such a value and reduce C using the same criterion.

Caching context embeddings in transformers A prominent architectural choice for foundation models is the *decoder-only transformer* [51], thanks to its auto-regressive nature and the ease of handling sequences of varying length either as input or output. This kind of architecture adopts

causal-attention, where past sequence tokens are not allowed to attend to future tokens. Thanks to this choice, once the key, value and query representations are computed for a token, these can be cached and re-used for future calculations, as the addition of a new subsequent token does not affect their value. However, most foundation models for time series adopt the common practice of *input patching*, where the individual input token to the model is not a single time-step, but a patch of L contiguous ones. This has been shown to provide several advantages [11]. However, this means that, in the standard setting where the input to the model consists of the most recent W observations, we cannot take advantage of caching, as all tokens change at each new time-step. In fact, to ensure past token would remain the same, we would have to perform inference only after L new observations are collected. In contrast to domain-specific global models, which are trained for a specific target application and thus could adapt the patch size L to the task's needs, foundation models cannot tune the patch size to any specific target application. If, similarly to Sec. 4, we provide as input a historical sequence of length C, followed by a shorter window of length W containing recent observations, we can take back the advantages of caching we lost due to input patching. In fact, as the historical sequence serves for GPI, it does not need to be updated for each new time-step, hence we can pre-compute all the relative key, value and query representations throughout all the layers in the transformer architecture. We can think of this collection of keys, values and queries as representing $\boldsymbol{e}^n$ in Eq. 8a. If q is the number of fixed tokens corresponding to the historical sequence, and $d \ll q$ is the number of those that are updated with recent observations, we can reduce the computational complexity from $\mathcal{O}\left((q+d)^2\right)$ to $\mathcal{O}\left(d(q+d)\right)$, as *attention* needs to be computed only between the new tokens and the historical tokens plus the new tokens. Clearly, here the computational advantage is enabled by the ability to keep the historical sequence, and its associated tokens, fixed.

J Related Works

Global models. Early works such as Salinas et al. [7] and Oreshkin et al. [52] demonstrated the effectiveness of deep learning for time series forecasting, showing that training a single predictor across a collection of time series can outperform specialized per-series models. Montero-Manso and Hyndman [9] further showed that such *global* models benefit from larger observation windows compared to *local* models, as they require higher capacity to match local model's performance. Attention-based models Vaswani et al. [51] were introduced [1, 10, 13, 12], to exploit the Transformer's ability of learning from long input sequences. Notably, Nie et al. [11] popularized the now-standard technique of slicing input sequences into *patches*, treating each patch as a token rather than individual observations. The superior performance of these architectures has been largely attributed to their ability to capture long-range dependencies, i.e., how distant past events influence future outcomes. More recent studies, however, have shown that other architectures such as linear models [33], MLPs [34], and *temporal convolutional network*s (TCNs) [35] can perform on par with transformers. Comprehensive reviews of deep learning models for time series forecasting can be found in Benidis et al. [2] and Wang et al. [3].

Foundation models. Recent years have seen the emergence of large-scale *foundation models* for time series, trained on massive datasets spanning multiple domains. These global models exhibit strong inductive forecasting capabilities, achieving robust performance on both time series observed and not observed during training. Zhou et al. [53] demonstrated that pre-trained large language models can be fine-tuned for time series forecasting, while later works explored training transformer architectures from scratch on extensive time series corpora. Rasul et al. [54] proposed a probabilistic decoder-only transformer, one of the few approaches that does not rely on input patching. Goswami et al. [16] introduced a unified backbone capable of addressing multiple time series tasks and efficiently adapting to new series via lightweight fine-tuning. The *Moirai* family [15, 28] further extended the line of probabilistic foundation models with encoder-only and decoder-only architectures. Several decoder-based models followed [14, 48, 29], targeting point prediction rather than probabilistic forecasting. Ansari et al. [49] drew inspiration from large language models, using a next-token prediction formulation by quantizing input patches and training via classification loss instead of regression. A detailed overview of these approaches can be found in Liang et al. [17].

In-context learning. In *large language model*s (LLMs), ICL refers to the ability to improve task performance when provided with relevant examples or descriptions as part of the input. Radford et al. [55] first demonstrated the benefits of including contextual documents, while subsequent

works [56, 27, 57] showed that textual task descriptions can further enhance performance. The *chain-of-thought* paradigm [58], where intermediate reasoning steps are reintroduced as input, represents another form of contextualization. Recent studies [59, 60, 61] have extended this principle to model training, yielding both scalability and performance gains. On the side of theoretical analysis, ICL has been linked to the ability of models trained on multiple tasks to implement a learning algorithm in their forward pass. In particular, Xie et al. [18], Zhang et al. [21] link this to Bayesian inference of latent variables characterizing the target task. Akyürek et al. [19] discuss how transformers can implement learning algorithms like gradient descent or ordinary least squares. Our results can be viewed through the lens of ICL: large input windows primarily supply contextual information to infer the target data-generating process, while only a smaller subset may directly influence the forecast. Recently, some works explored ICL in the context of time series forecasting. From a theoretical perspective, Li et al. [20] provide a mechanistic analysis of ICL as function learning. Their formulation includes 1-step-ahead prediction of contiguous observations from dynamical systems, and they derive generalization bounds for the task. On the methodological side, Faw et al. [38] showed that sequences from related series can enhance forecasting accuracy by enriching the model's context. Similarly, Auer et al. [39] and Ansari et al. [23] showed how covariate signals, i.e., time series correlated to the target one, can improve accuracy, and assigned this benefit to the additional information to contextualize the task. Lu et al. [41] showed one can restructure the input to a forecasting model as a series of input/output examples plus an input query and achieve better predictive performance. Williams et al. [40], instead, showed that textual descriptions can be used as guiding context to enhance forecasting accuracy. For a comprehensive review of ICL in LLMs, see Dong et al. [22].

Global-local models. When the set of target time series is fixed, *global-local* models are an effective strategy to improve performance and lessen the computational cost, w.r.t. global models, by combining shared global components with series-specific local ones [44]. Nie et al. [11] highlighted the advantages of series-specific learnable positional encodings, while Cini et al. [43] demonstrated the effectiveness of learnable node embeddings in graph-based spatiotemporal forecasting. Top entries in the M4 competition [8] similarly employed global-local designs: Smyl [42] integrated shared architectures with series-specific preprocessing parameters, and Montero-Manso et al. [62] leveraged handcrafted features as local inputs. Under our formulation, in purely global models, extending the input sequence can help reduce the uncertainty about the target data-generating process. In inductive settings this can improve the model's performance, playing a similar role to that of local components in transductive settings.

Adaptive predictors. Our observations suggest that foundation models for time series should integrate two complementary components: one to infer contextual information about the observed series, and another to leverage this information for prediction. This idea echoes the concept of *adaptive neural networks*, where one sub-network modulates another's weights or representations based on the input. Examples include *fast weight programmers* [63] and *hypernetworks* [64]. Oreshkin et al. [65] theorized that, in deep forecasting architectures, early layers may act as a meta-learning outer loop [66], adapting representations for later predictive layers. Although meta-learning strategies have been explored to a limited extent in time series forecasting, e.g., Talagala et al. [67] used handcrafted features for model selection, and Norton et al. [68] introduced a *signal-mapper* to predict model parameters in low-data regimes, scalable adaptive predictors for large-scale foundation models remain largely unexplored.

K Additional results

This section provides additional or complementary results for the experiments in the main paper.

K.1 Complementary results for Sec. 3.1

In Fig. 13 we visualize example predictions produced by the pre-trained foundation models considered in Fig. 2, i.e., Moirai2 [28] and TimerXL [29], on two real-world environments, i.e., LargeST-SD and ClmUS-T. The figure visually compares the results obtained by using, as input, a standard window of length $W = 720$, immediately preceding the target time-step t, with those obtained by concatenating a historical sequence of length $C = 720$ and a short window of length $W = 24$. We

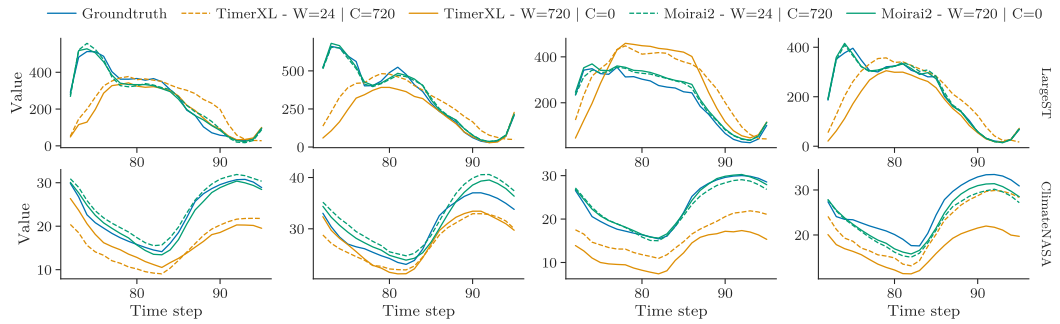
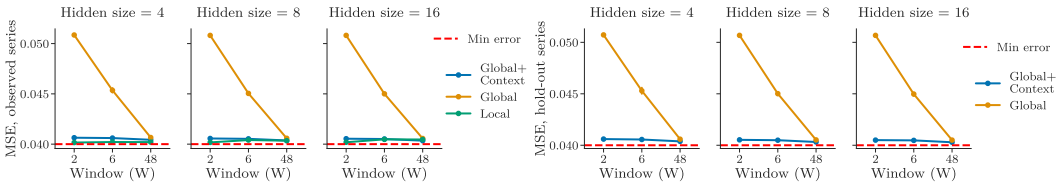


Figure 13: Example trajectories produced by foundation models (color-coded), on real-world datasets, when given as input the standard window o length W (solid line) or a short window of plus context (a historical sequence) of length C (dashed line).



(a) Transductive setting

(b) Inductive setting

Figure 14: 1-step-ahead forecasting error (MSE, 3 runs $\pm std$) for NAR(2). $C = 46$. Subplots correspond to different GRU hidden size. *Min error* refers to the known irreducible MSE.

can observe how, in general, predictions have similar qualities. In particular we can notice that using the historical sequence sometimes leads to predictions that seem shifted upwards/downwards w.r.t. to those obtained with the standard window. As these models expect temporally contiguous input sequences, this effect might be the result of the artifact introduced by concatenating two sequences with a temporal gap between them.

K.2 Complementary results for 3.2

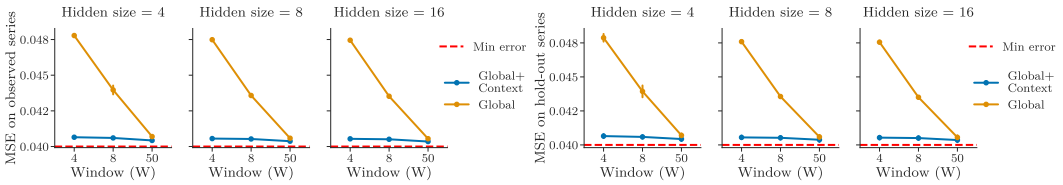
In Fig. 14 we report complementary results to those in Fig. 3, showing how changing the model's hidden size has no significant effect on the results. In particular, Fig. 14a, reports the error obtained on the future realizations of the time series observed during training (transductive setting). We can observe similar results to those obtained on the entirely held-out time series (inductive setting), which are reported in Fig. 14b.

In addition, Fig. 15 and Fig. 16 report results when repeating the experiment with NAR processes with order $P = 4$ and $P = 6$ respectively, where we omitted local models for simplicity. The evaluated window lengths W were incremented according to the process increased temporal order, to avoid providing the model with less than P observations as input. We can observe similar results to the case $P = 2$, where providing the additional context sequence improves the model performance already when $W = P$. Notably, in this case, the predictors gap from the minimum attainable error sees a relative increase with the increment of P, possibly stemming from the need for additional input observations to infer the process generating the input.

K.3 Complementary results for Sec. 3.3

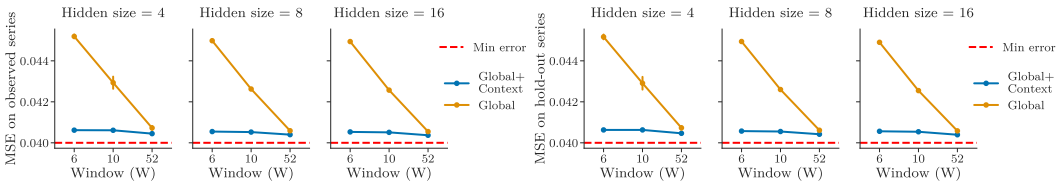
Fig. 17 reports complementary w.r.t. Fig. 4. While the latter shows the error on held-out series (inductive setting), the former shows the error on the future, unobserved, realizations of the time series used for training (transductive setting). Overall, we can observe a similar behavior between the two different testing scenarios.

To complement results from the experiment in Fig. 5, we compare, across the 4 considered real-world datasets, the foundation models' error at different window lengths W, against that of additional



(a) Transductive setting (b) Inductive setting

Figure 15: 1-step-ahead forecasting error (MSE, 3 runs $\pm std$) for NAR(4). $C = 46$. Subplots correspond to different GRU hidden size. *Min error* refers to the known irreducible MSE.



(a) Transductive setting (b) Inductive setting

Figure 16: 1-step-ahead forecasting error (MSE, 3 runs $\pm std$) for NAR(6). $C = 46$. Subplots correspond to different GRU hidden size. *Min error* refers to the known irreducible MSE.

domain-specific baselines, i.e., ModernTCN, TSMixer, and LRU. Tab. 6 reports the results of the experiment, which follows the same protocol of Fig. 5, where models are evaluated *inductively*, i.e., on time series that were entirely held-out during training. We focus on smaller values of W, where the highest error gap is expected. Results are consistent with our observations in the main text, domain-specific models have better input length to error trade-offs compared to foundation models that are trained on heterogeneous time series datasets.

K.4 Complementary results for Sec. 4

Here, we report complementary results showing the error on the future, unobserved, realizations of the time series used for training (transductive setting), whereas the main paper reports the error on held-out series (inductive setting). In particular, Fig. 18 is complementary to Fig. 7. We can see the presence of similar trends, further validating our analysis in the main paper.

In addition, we report expanded variants of Fig. 18 and Fig. 7, respectively in Fig. 20 and Fig. 19. These larger figures report separate sub-plots for each model and dataset pair, in order to enhance readability.

Similarly, to complement Fig. 8, we report performance-cost Pareto frontiers for individual architectures and datasets. Fig. 21 reports FLOPs, Fig. 22 inference time on CPU, Fig. 23 inference time on GPU and Fig. 24 the peak GPU memory occupancy. Notably, results are similar to the aggregated plot in Fig. 8. In most scenarios the decoupled approach dominates the standard one at lower computational costs, which correspond to shorter input windows. As one would expect, given our analysis on the effect of GPI, at higher costs (larger W) the Pareto frontiers often align.

Speedup within accuracy loss budget To complement the reported Pareto frontiers, we also provide a different perspective on the performance-cost trade-off. For each dataset and model reported in Fig. 7, we consider, as reference, the minimum MAE achieved by the *standard* approach across different input lengths W. Then, for both the standard and *decoupled* approach, we compute the attainable improvement factors in computational performance when allowing a penalty of up to 10% of the reference MAE. Tab. 7 reports the computational improvement factors in terms of FLOPs, inference time on CPU, inference time on GPU and peak GPU memory occupancy. Notably, in most scenarios the decoupled approach allows significant scalability within this 10% accuracy budget.

Additional time series benchmarks We conduct experiments comparing the standard and decoupled approach introduced in Sec. 4 on the *electricity*, *traffic* and *weather* datasets [1], considering the case where both the *base model* (Eq. 8b) and the *embedding module* (Eq. 8a) are implemented by

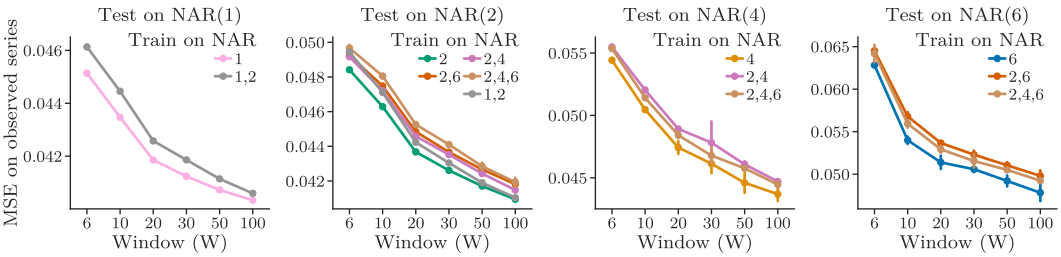


Figure 17: 1-step-ahead forecasting (MSE, *transductive*, 3 runs $\pm std$) for different combinations of NAR domains. Columns show error on a specific domain, color identifies the training domains.

Table 6: 24-step ahead forecasting error (MAE, *inductive*, 3 runs $\pm std$) against window length W. Foundation models (F) against domain-specific global models (S). **Bold** denotes domain-specific models that outperform *all* foundation models, underline denotes those that outperform *some*.

Dataset	W	Foundation (F)				Specialized (S)		
		Chronos2	Moirai2	TimerXL	TimesFM	LRU	ModernTCN	TSMixer
LargeST-SD	24	47.219±0.998	103.168±5.574	61.255±1.624	50.170±1.503	**31.758**±0.748	**26.394**±0.841	**35.595**±1.439
	72	32.206±0.924	80.484±2.742	41.853±0.382	30.204±0.760	**25.866**±1.113	**21.165**±1.003	**28.427**±1.158
CEREn	24	0.940±0.034	1.593±0.116	1.132±0.034	0.956±0.049	**0.801**±0.023	**0.793**±0.022	**0.803**±0.031
	72	0.839±0.037	1.521±0.157	0.962±0.030	0.732±0.029	0.777±0.042	0.759±0.035	0.772±0.029
ClmUS-T	24	2.646±0.037	10.439±0.128	4.106±0.052	2.841±0.044	**1.853**±0.026	**1.773**±0.028	**1.981**±0.033
	72	2.265±0.039	11.339±0.016	3.160±0.048	2.087±0.034	**1.766**±0.021	**1.729**±0.040	**1.833**±0.033
ClmUS-R	24	54.806±0.024	145.611±0.747	91.976±0.478	118.220±1.020	**38.088**±0.456	**35.526**±0.816	**41.659**±0.811
	72	43.187±0.135	69.532±1.563	65.570±0.475	48.303±0.236	**37.582**±0.621	**35.275**±0.353	**37.946**±0.352

PatchTST. Fig. 25 reports the results for the transductive (Fig. 25a) and inductive (Fig. 25b) settings. Results are consistent with our analysis and the results observed in Fig. 7. The advantage provided by the decoupled approach depends on the target time series distribution, e.g., on the *electricity* dataset the advantage is only for the smallest input window considered, possibly as few observations are already sufficient for GPI in this task. Note that for *weather* we report only the transductive performance as the dataset involves only 21 time series measuring different weather variables, hence the data are not appropriate to consider a hold-out set on which to measure inductive performance.

Effect of parameter count The decoupled approach in Sec. 4 involves adding an *embedding module* (Eq. 8a) to a *base model* (Eq. 8b). This results in an overall architecture with a higher parameter count, compared to the *base model* alone. To validate that simply increasing the parameter count of the *base model* would not allow to close the performance gap at shorter window lengths W observed in Fig. 7, we train a PatchTST model with doubled hidden size (roughly matching the parameter count of the end-to-end decoupled architecture). Tab. 8 reports the obtained results, comparing the standard PatchTST with doubled hidden size (2xHS), to the standard and decoupled PatchTST with the regular hidden size used in Fig. 7 (1xHS). As the results show, while doubling the hidden size (2xHS, standard) may result in an error improvement compared to the baseline (1xHS, standard), it does not allow to close the gap with the decoupled approach (1xHS, decoupled), even though it approximately matches its parameter count.

Effect of embedding module architecture To validate that the results in Sec. 4 are not due to the choice of using PatchTST as the architecture for the *embedding module*, we experiment with implementing $G(\cdot)$ (Eq. 8a) by means of a MLP. Tab. 9 reports the results obtained when training decoupled models in which $F(\cdot)$ (Eq. 8b) is implemented by PatchTST and $G(\cdot)$ by a MLP. We consider the four real-world datasets of Sec. 4 and small values of W. The table compares the MLP embedding module (i.e., G=MLP) to the PatchTST one (i.e., G=PatchTST) and to the standard approach where $G(\cdot)$ is not present (i.e., G=None). The results show how implementing $G(\cdot)$ by means of a MLP does not affect our conclusions. Indeed, the performance is close to that of the

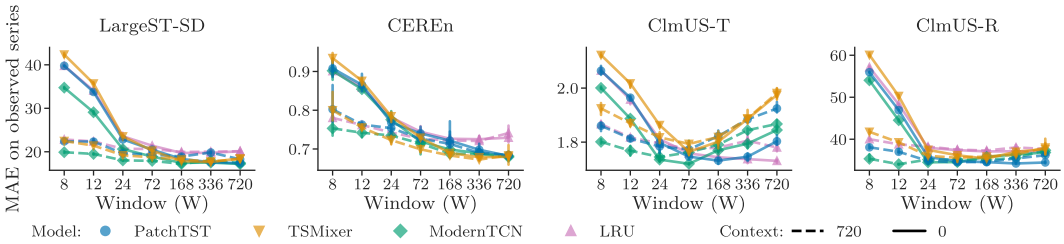


Figure 18: 24-step-ahead forecasting error (MAE, *transductive*, 3 runs $\pm std$). Color/marker denotes the model. Line-style denotes the approach: decoupling (dashed) or standard (solid).

Table 7: Computational advantage for standard and decoupled approaches when allowing up to 10% relative MAE performance cost w.r.t. the best model using the standard approach. The table shows the speedup factors for CPU time, CUDA time, FLOPs, and peak CUDA memory. Best results are in bold.

Model	Dataset	Approach	CPU T (×)	CUDA T (×)	FLOPs (×)	CUDA Mem. (×)
LRU	CEREn	Standard	19.88	4.95	13.98	13.94
		Decoupled	**33.50**	**7.43**	**27.05**	**27.62**
	ClmUS-R	Standard	8.00	2.61	6.99	6.97
		Decoupled	**18.69**	**2.73**	**19.93**	**20.01**
	ClmUS-T	Standard	43.61	21.70	29.96	29.85
		Decoupled	**101.91**	**22.69**	**85.41**	**85.71**
	LargeST-SD	Standard	2.94	1.64	2.33	2.32
		Decoupled	**8.86**	**2.32**	**6.88**	**6.94**
ModernTCN	CEREn	Standard	8.55	6.96	**8.06**	**9.45**
		Decoupled	**9.39**	**7.14**	7.32	9.43
	ClmUS-R	Standard	1.00	1.00	1.00	1.00
		Decoupled	**1.21**	**3.51**	**1.09**	**2.13**
	ClmUS-T	Standard	1.98	1.08	2.98	2.74
		Decoupled	**2.39**	**3.78**	**3.25**	**5.85**
	LargeST-SD	Standard	4.66	**11.00**	3.92	4.19
		Decoupled	**16.00**	10.51	**21.85**	**25.75**
PatchTST	CEREn	**Standard**	**15.49**	8.41	**15.62**	**9.02**
		Decoupled	12.94	**13.38**	15.49	8.96
	ClmUS-R	Standard	14.22	**4.87**	17.95	11.11
		Decoupled	**48.36**	2.10	**49.25**	**21.54**
	ClmUS-T	Standard	4.96	**3.13**	7.91	5.62
		Decoupled	**15.24**	2.95	**22.32**	**11.20**
	LargeST-SD	**Standard**	**8.09**	**9.28**	**6.18**	**4.18**
		Decoupled	7.61	9.01	6.16	4.17
TSMixer	CEREn	Standard	5.88	3.17	14.56	4.49
		Decoupled	**29.11**	**4.94**	**37.64**	**9.82**
	ClmUS-R	**Standard**	12.76	**3.39**	**20.18**	**6.29**
		Decoupled	**12.90**	2.50	10.68	4.96
	ClmUS-T	Standard	4.90	**2.12**	4.88	2.78
		Decoupled	**9.72**	**2.12**	**8.21**	**5.42**
	LargeST-SD	Standard	2.26	1.98	3.52	1.98
		Decoupled	**29.11**	**4.94**	**37.64**	**9.82**

case where $G(\cdot)$ is implemented by PatchTST, and the performance gap compared to the standard approach is preserved.

K.5 Complementary results for Example E.1

To complement the results shown in Fig. 9 we provide, in Figures 26, 27, 28 and 29, results obtained under different data generation hyper-parameters. Specifically we change the random seed to sample the two AR(2) process parameters, the value of σ^2, and the number of generated time steps. Results are similar, showing alignment between the learned neural network predictions and those of the closed-form optimal predictor for an input window of length $W = 2$.

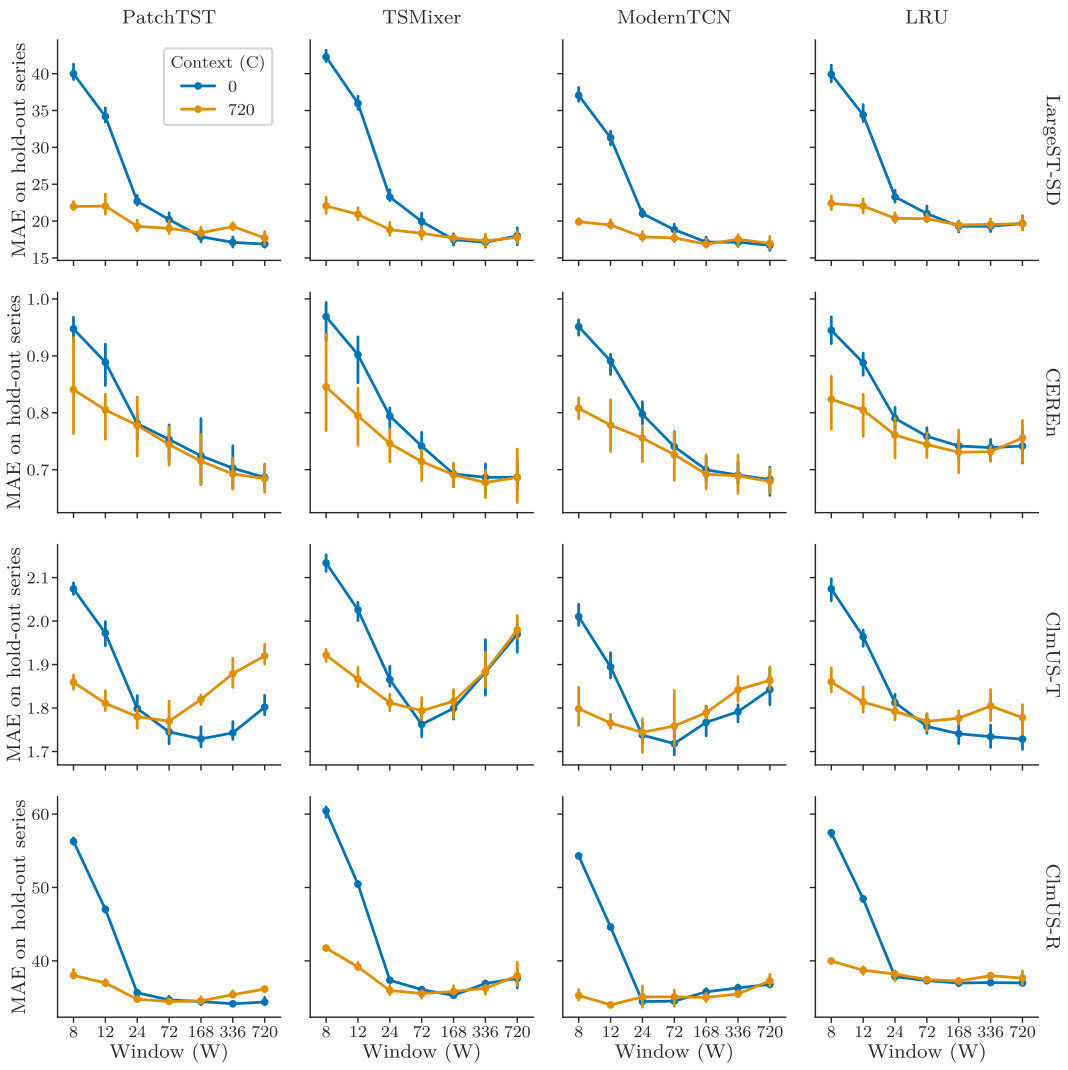


Figure 19: 24-step-ahead forecasting error (MAE, *inductive*, 3 runs $\pm std$). Columns correspond to different models while rows represent different datasets. Line-color denotes decoupling (orange) or not (blue).

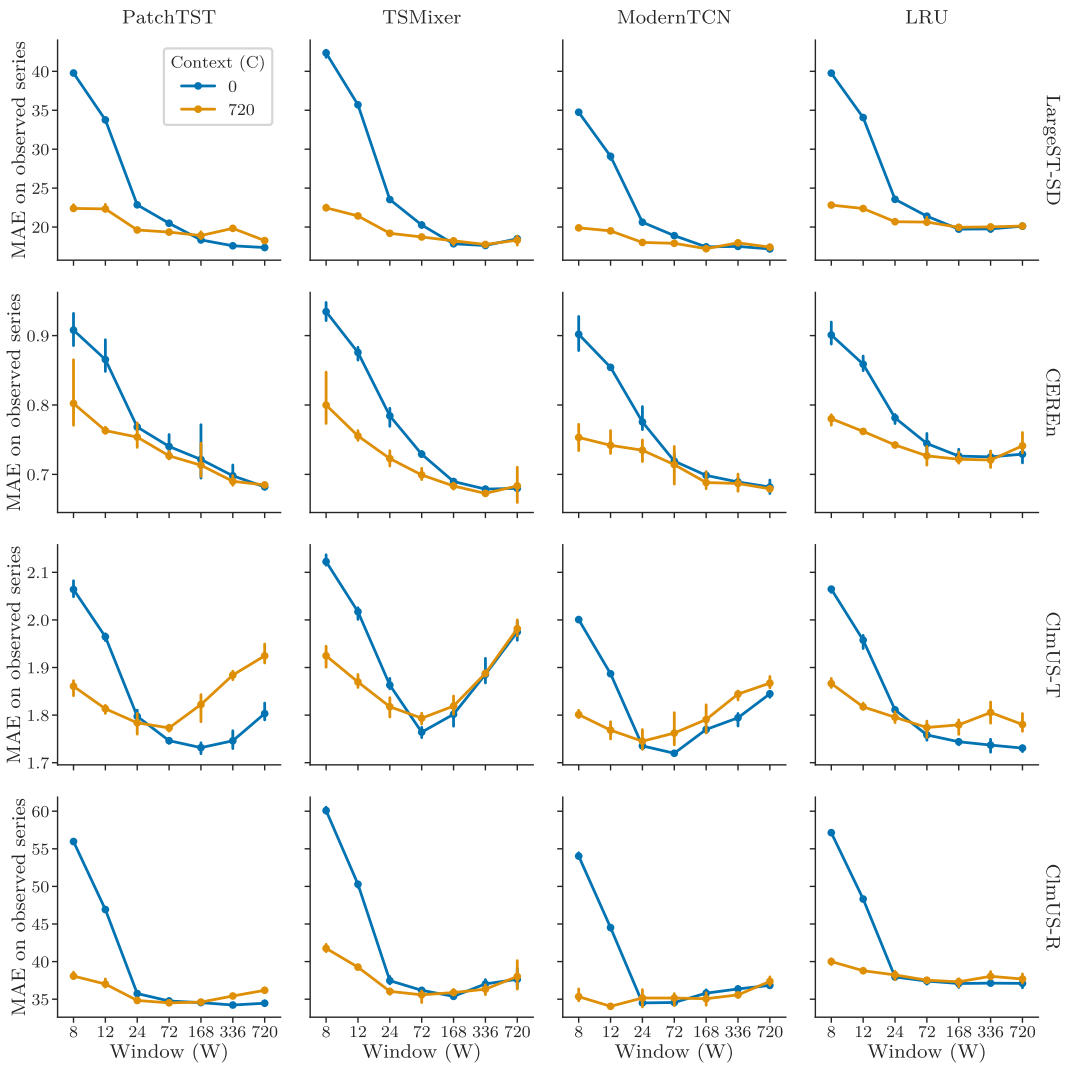


Figure 20: 24-step-ahead forecasting error (MAE, *transductive*, 3 runs $\pm std$). Columns correspond to different models while rows represent different datasets. Line-color denotes decoupling (orange) or not (blue).

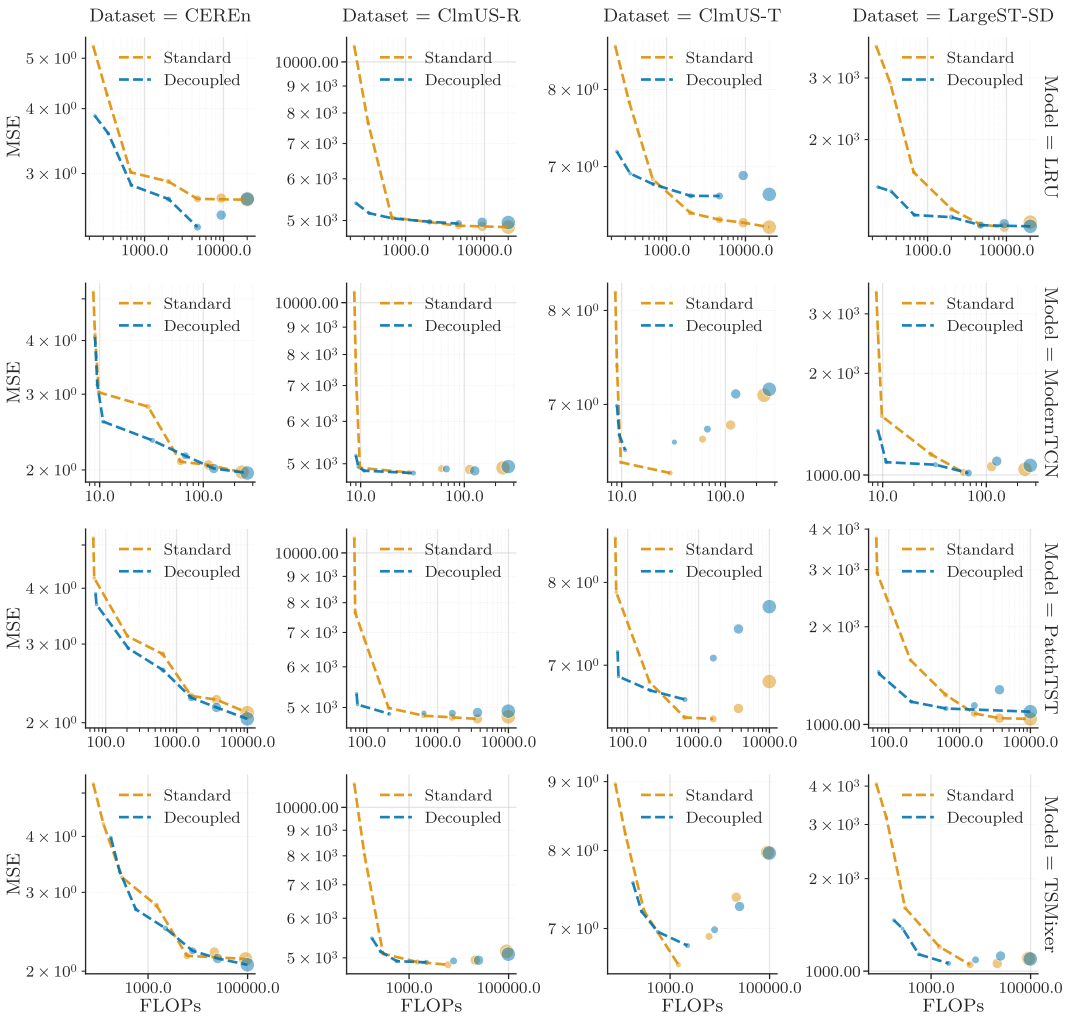


Figure 21: Pareto frontiers, *inductive* MSE (y-axis) vs FLOPs (MFLOPs, x-axis), for the decoupled (blue) and standard (orange) approach. Dots correspond to model/dataset pairs from Fig. 7 (3 runs ±std). Dot size is proportional to the input window length W. Rows correspond to different models, columns to different datasets.

Table 8: 24-step ahead forecasting error (MAE, inductive, 3 runs $\pm std$). (1xHS, standard) and (1xHS, decouple) denote the standard and decoupled approach in Fig. 7. (2xHS, standard) denotes the same training conditions of the standard approach but all layers' hidden sizes are doubled.

Dataset	**Model**	**W = 8**	**W = 12**
LargeST-SD	PatchTST (1xHS, standard)	39.995 ± 1.128	34.188 ± 1.027
	PatchTST (2xHS, standard)	37.933 ± 1.164	32.591 ± 1.085
	PatchTST (1xHS, decoupled)	21.980 ± 0.611	22.032 ± 1.463
CEREn	PatchTST (1xHS, standard)	0.947 ± 0.021	0.889 ± 0.037
	PatchTST (2xHS, standard)	0.945 ± 0.023	0.872 ± 0.021
	PatchTST (1xHS, decoupled)	0.841 ± 0.086	0.805 ± 0.045
ClmUS-T	PatchTST (1xHS, standard)	2.074 ± 0.014	1.972 ± 0.028
	PatchTST (2xHS, standard)	2.033 ± 0.020	1.926 ± 0.024
	PatchTST (1xHS, decoupled)	1.859 ± 0.016	1.811 ± 0.026
ClmUS-R	PatchTST (1xHS, standard)	56.277 ± 0.442	47.026 ± 0.355
	PatchTST (2xHS, standard)	54.626 ± 0.580	45.617 ± 0.384
	PatchTST (1xHS, decoupled)	38.033 ± 0.699	36.980 ± 0.437

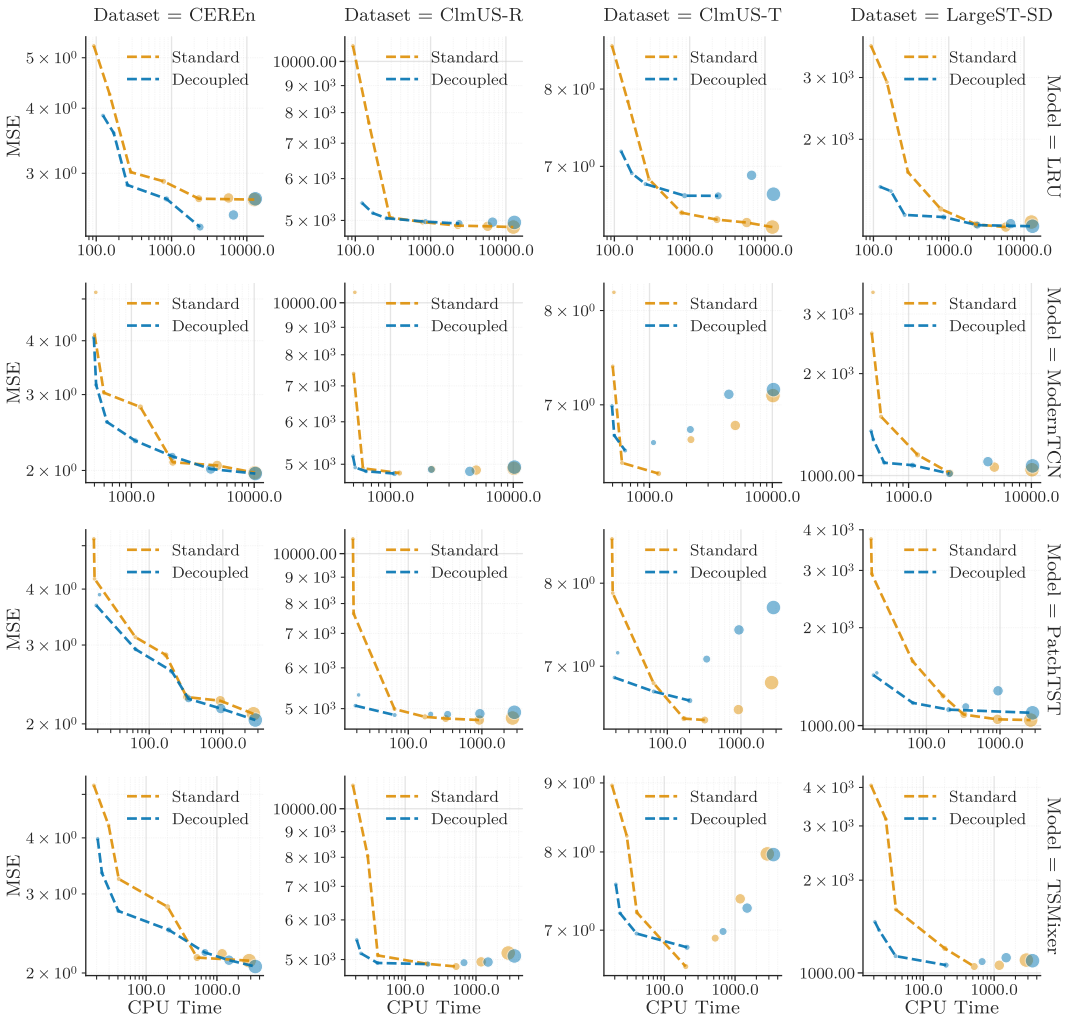


Figure 22: Pareto frontiers, *inductive* MSE (y-axis) vs CPU inference time (ms, x-axis), for the decoupled (blue) and standard (orange) approach. Dots correspond to model/dataset pairs from Fig. 7 (3 runs ±std). Dot size is proportional to the input window length W. Rows correspond to different models, columns to different datasets.

Table 9: 24-step ahead forecasting error (MAE, inductive, 3 runs $\pm std$). (G=None) and (G=PatchTST) denote the standard and decoupled approach in Fig. 7. (G=MLP) denotes the same training conditions of the decoupled approach but $G(\cdot)$ is implemented by a MLP.

Dataset	**Model**	**W = 8**	**W = 12**
LargeST-SD	PatchTST (G=None)	39.995 ± 1.128	34.188 ± 1.027
	PatchTST (G=MLP)	23.384 ± 0.719	22.790 ± 1.058
	PatchTST (G=PatchTST)	21.980 ± 0.611	22.032 ± 1.463
CEREn	PatchTST (G=None)	0.947 ± 0.021	0.889 ± 0.037
	PatchTST (G=MLP)	0.823 ± 0.039	0.810 ± 0.035
	PatchTST (G=PatchTST)	0.841 ± 0.086	0.805 ± 0.045
ClmUS-T	PatchTST (G=None)	2.074 ± 0.014	1.972 ± 0.028
	PatchTST (G=MLP)	1.863 ± 0.028	1.821 ± 0.016
	PatchTST (G=PatchTST)	1.859 ± 0.016	1.811 ± 0.026
ClmUS-R	PatchTST (G=None)	56.277 ± 0.442	47.026 ± 0.355
	PatchTST (G=MLP)	38.132 ± 0.026	37.013 ± 0.439
	PatchTST (G=PatchTST)	38.033 ± 0.699	36.980 ± 0.437

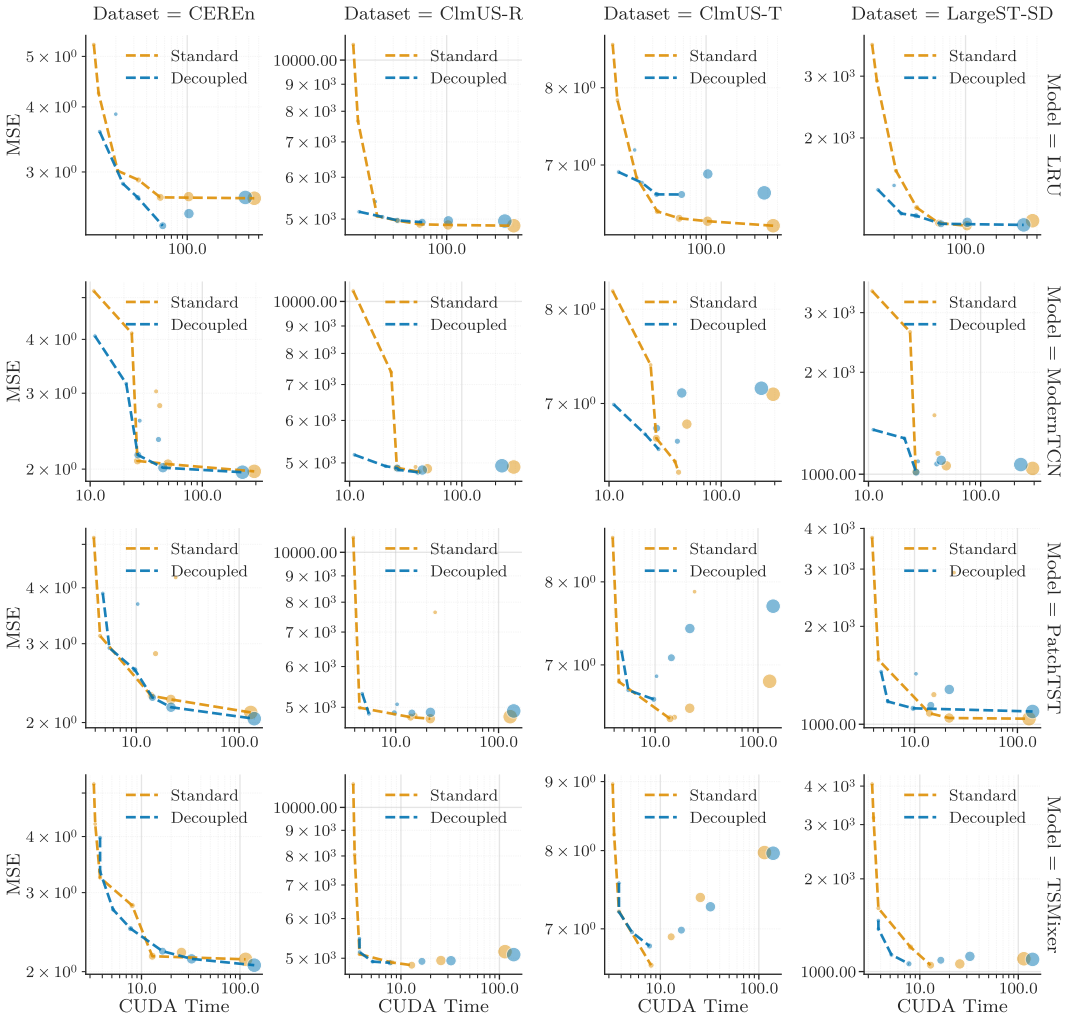


Figure 23: Pareto frontiers, *inductive* MSE (y-axis) vs GPU inference time (ms, x-axis), for the decoupled (blue) and standard (orange) approach. Dots correspond to model/dataset pairs from Fig. 7 (3 runs ±std). Dot size is proportional to the input window length W. Rows correspond to different models, columns to different datasets.

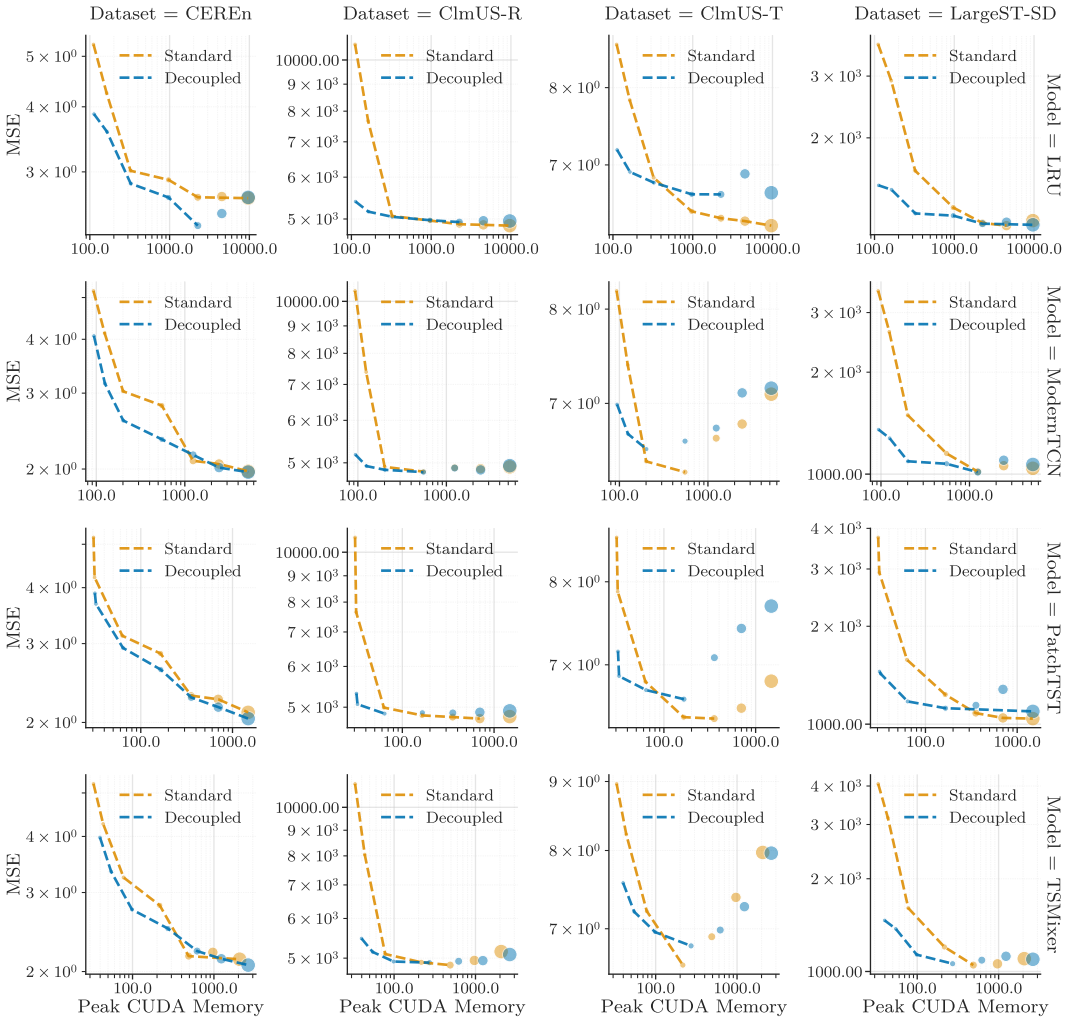
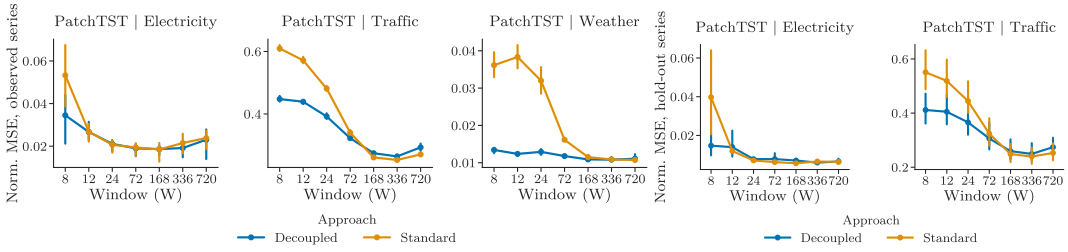


Figure 24: Pareto frontiers, *inductive* MSE (y-axis) vs peak GPU memory occupancy (MB, x-axis), for the decoupled (blue) and standard (orange) approach. Dots correspond to model/dataset pairs from Fig. 7 (3 runs ±std). Dot size is proportional to the input window length W. Rows correspond to different models, columns to different datasets.



(a) Transductive setting

(b) Inductive setting

Figure 25: 96-step-ahead forecasting error (normalized MSE, 3 runs $\pm std$) for *electricity*, *traffic* and *weather* datasets. Standard (orange) vs decoupled (blue) approach. $C = 720$.

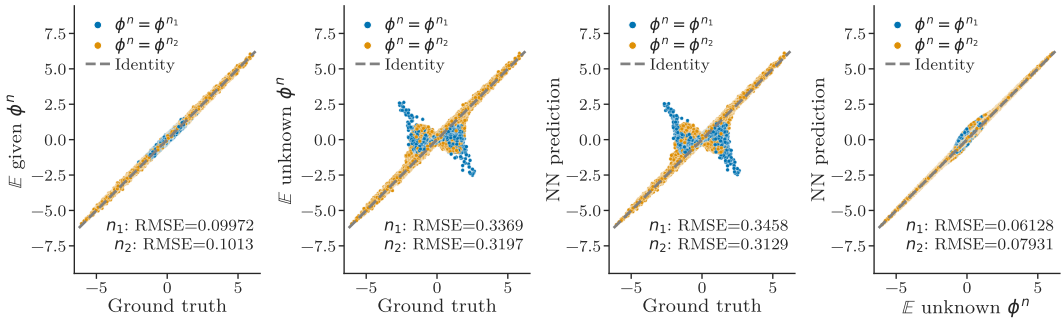


Figure 26: Parity plots for Example E.1. Data generation hyper-parameters: $\sigma^2 = 0.1$, $seed = 19$, $\#timesteps = 100000$. *Ground truth* denotes the actual realizations from the stochastic process. $\mathbb{E}$ *given* $\boldsymbol{\phi}^n$ denotes the expected value for the future observation when knowing the process parameters and the last 2 observations. $\mathbb{E}$ *unknown* $\boldsymbol{\phi}^n$ denotes the predictions obtained by using the real process parameters weighted by the likelihood in Eq. 19 computed in closed form. *NN prediction* denotes the predictions obtained by a neural network optimized to minimize the MSE. The RMSE is computed between the two variables on each plot's axes, and reported separately for processes with $\boldsymbol{\phi}^n = \boldsymbol{\phi}^{n_1}$ and $\boldsymbol{\phi}^n = \boldsymbol{\phi}^{n_2}$.

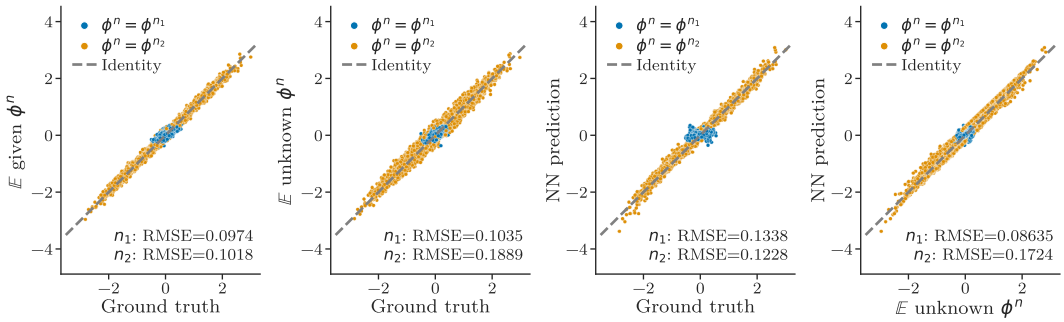


Figure 27: Parity plots for Example E.1. Data generation hyper-parameters: $\sigma^2 = 0.1$, $seed = 42$, $\#timesteps = 10000$. *Ground truth* denotes the actual realizations from the stochastic process. $\mathbb{E}$ *given* $\boldsymbol{\phi}^n$ denotes the expected value for the future observation when knowing the process parameters and the last 2 observations. $\mathbb{E}$ *unknown* $\boldsymbol{\phi}^n$ denotes the predictions obtained by using the real process parameters weighted by the likelihood in Eq. 19 computed in closed form. *NN prediction* denotes the predictions obtained by a neural network optimized to minimize the MSE. The RMSE is computed between the two variables on each plot's axes, and reported separately for processes with $\boldsymbol{\phi}^n = \boldsymbol{\phi}^{n_1}$ and $\boldsymbol{\phi}^n = \boldsymbol{\phi}^{n_2}$.

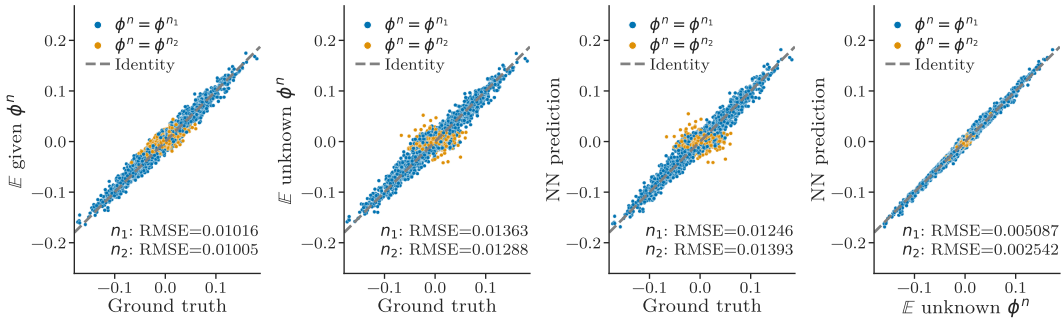


Figure 28: Parity plots for Example E.1. Data generation hyper-parameters: $\sigma^2 = 0.01$, $seed = 7$, $\#timesteps = 10000$. *Ground truth* denotes the actual realizations from the stochastic process. $\mathbb{E}$ *given* $\boldsymbol{\phi}^n$ denotes the expected value for the future observation when knowing the process parameters and the last 2 observations. $\mathbb{E}$ *unknown* $\boldsymbol{\phi}^n$ denotes the predictions obtained by using the real process parameters weighted by the likelihood in Eq. 19 computed in closed form. *NN prediction* denotes the predictions obtained by a neural network optimized to minimize the MSE. The RMSE is computed between the two variables on each plot's axes, and reported separately for processes with $\boldsymbol{\phi}^n = \boldsymbol{\phi}^{n_1}$ and $\boldsymbol{\phi}^n = \boldsymbol{\phi}^{n_2}$.

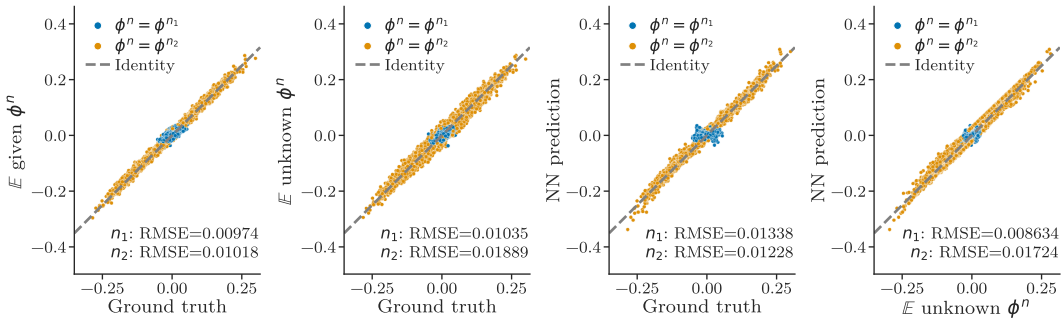


Figure 29: Parity plots for Example E.1. Data generation hyper-parameters: $\sigma^2 = 0.01$, $seed = 42$, $\#timesteps = 10000$. *Ground truth* denotes the actual realizations from the stochastic process. $\mathbb{E}$ *given* $\boldsymbol{\phi}^n$ denotes the expected value for the future observation when knowing the process parameters and the last 2 observations. $\mathbb{E}$ *unknown* $\boldsymbol{\phi}^n$ denotes the predictions obtained by using the real process parameters weighted by the likelihood in Eq. 19 computed in closed form. *NN prediction* denotes the predictions obtained by a neural network optimized to minimize the MSE. The RMSE is computed between the two variables on each plot's axes, and reported separately for processes with $\boldsymbol{\phi}^n = \boldsymbol{\phi}^{n_1}$ and $\boldsymbol{\phi}^n = \boldsymbol{\phi}^{n_2}$.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction reflect the main claims and contributions of our work. In particular, when listing contributions in the introduction (Sec. 1), we explicitly reference the sections of the paper that are relevant to each contribution.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are explicitly discussed in the conclusions (Sec. 6) and throughout the document where appropriate. Throughout the paper we explicitly state the assumptions that are relevant to our work (e.g., in Sec. 2 and Sec. 3).

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Proposition 3.2 references the relevant assumptions (e.g., Eq. 6, Assumption 3.1), and a full proof is provided in App. D.1.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The appendix provides all relevant information to ensure reproducibility. References to the considered datasets are provided in App. G, together with information about the synthetically generated settings in App. G.1. Information about data splitting is in App. G.2. The considered architectures are referenced and/or described in App. H.1.2 and App. H.1.1, while the hyper-parameters and training details are reported in App. H.2. Moreover, code for reproducibility, including the seeds controlling randomized operations, is provided to the reviewers and will be released upon publication. A reproducibility statement is provided in App. A.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Code to reproduce the experiments is provided to the reviewers and will be openly released upon publication. The code is complemented by a README explaining how to reproduce the main experiments. Environment and experiment configuration files are provided in the code to ensure reproducibility. The considered datasets are either included or relevant references are provided. As detailed in the reproducibility statement (App. A), the CEREn dataset is not openly available, and a request for data access has to be made to the appropriate authority (see CER [31]). In addition, App. H.2 provides information about the hardware requirements to reproduce the experiments and the expected runtime.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Details about the experimental settings are provided throughout App. G and App. H. These include details about data splitting and the *transductive*/*inductive* performance testing (App. G.2), as well as details about synthetic data generation (App. G.1), hyperparameters, loss functions, and optimizers (App. H.2).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.

- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Experiments are repeated across multiple runs with different (fixed) seeds controlling random operations to ensure reproducibility. Figures and tables report the mean and standard deviation across runs (either via error bars or in text), specifying in the caption the number of runs the statistics come from.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Details about the computational resources required to reproduce the experiments are reported in App. H.2.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

Justification: This research does not involve human subjects nor data collection. Datasets used in this work are either openly available or used in accordance with their terms of use to the best of our knowledge. We do not foresee significant harmful societal impact as an immediate or direct consequence of our research, as discussed in the broader impact statement (App. B).

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: Possible positive and negative societal impacts of this work are discussed in the broader impact statement (App. B). However, given the nature of this work, we do not foresee significant and/or direct harmful societal impact.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: This work does not involve the release of new datasets, and the risk of misuse, for models that can be trained as a result of open code release, is in line with that of other existing models.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: For the architectures considered in the paper, and the pre-trained foundation models, we reference the original papers. The sources for the considered datasets are referenced in the text, and App. G mentions the corresponding licenses where applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: While we provide code for reproducibility, this work does not release new assets in the sense of datasets or pre-trained models or code libraries. Nonetheless, the code for reproducibility is accompanied by a README file which explains the necessary steps to reproduce the experiments, and the code will be made publicly available upon publication.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing, neither it involves research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing, neither it involves research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: This paper does not involve LLMs, neither as a component of any proposed methodology, nor as part of the experiments.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.