TACTiS: Transformer-Attentional Copulas for Time Series

Alexandre Drouin [*1] **Étienne Marcotte** [*1] **Nicolas Chapados** [*1]

Abstract

The estimation of time-varying quantities is a fundamental component of decision making in fields such as healthcare and finance. However, the practical utility of such estimates is limited by how accurately they quantify predictive uncertainty. In this work, we address the problem of estimating the joint predictive distribution of high-dimensional multivariate time series. We propose a versatile method, based on the transformer architecture, that estimates joint distributions using an attention-based decoder that provably learns to mimic the properties of non-parametric copulas. The resulting model has several desirable properties: it can scale to hundreds of time series, supports both forecasting and interpolation, can handle unaligned and non-uniformly sampled data, and can seamlessly adapt to missing data during training. We demonstrate these properties empirically and show that our model produces state-of-the-art predictions on multiple real-world datasets.

1. Introduction

In numerous time series forecasting contexts, data presents itself in a raw form that rarely matches the standard assumptions of classical forecasting methods. For instance, in healthcare settings and economic forecasting, groups of related time series can have different sampling frequencies, be sampled irregularly, and exhibit missing values (Shukla & Marlin, 2021b; Sun et al., 2020). Covariates that are predictive of future behavior may not be available for all historical data, or may be available in a different form, e.g., due to changes in measurement methodology. Moreover, optimal decision-making in downstream tasks generally requires the full joint predictive distribution over arbitrary future time horizons (Peterson, 2017), not just marginal quantiles thereof at a fixed horizon. We seek to develop general forecasting methods that are both suitable for a wide range of downstream tasks and that can handle all stylized facts about real-world time series—as they are, not as they ought to be—namely:

- Joint characterization of a multivariate stochastic process, forecasting trajectories at arbitrary time horizons;
- Presence of non-stochastic covariates, either static or time-varying, used as conditioning variables;
- Variables within the process that are measured at different sampling frequencies or irregularly sampled;
- Missing values for arbitrary time points and variables;
- Variables with different domains—$\mathbb{R}$, $\mathbb{R}^+$, $\mathbb{N}$, $\mathbb{N}^+$, $\mathbb{Z}$—with skewed and fat-tailed marginal behavior.

Classical times series models, such as ARIMA (Box et al., 2015) and exponential smoothing methods (Hyndman et al., 2008), are very restricted in their handling of the above stylized facts. Although extensions have been proposed that deal with individual issues, they cannot easily deal with all and require considerable domain knowledge to be effective. Machine learning models have recently gained in popularity (Benidis et al., 2020). Nevertheless, methods introduced in recent years all suffer from limitations when dealing with one or more of the above listed stylized facts, or neglect in their handling of the full predictive distribution.

Separately, multivariate forecasting models based on copulas have been popular in econometrics for more than a decade (Patton, 2012; Rémillard et al., 2012; Krupskii & Joe, 2020; Mayer & Wied, 2021). These models enable the separate characterization of the joint behavior of a group of random variables from their marginal behavior. This has been found to be especially valuable in areas such as finance and insurance where marginals are known to exhibit particular patterns of skewness and kurtosis. Recently in machine learning, low-rank Gaussian copula processes with LSTMs (Hochreiter & Schmidhuber, 1997) have been proposed for high-dimensional forecasting (Salinas et al., 2019).

Building on the recent successes of transformers as general-purpose sequence models (Vaswani et al., 2017) and their success in time series forecasting (Rasul et al., 2021b; Tashiro et al., 2021; Tang & Matteson, 2021), we propose a transformer architecture that can tackle all the above stylized facts about real-world time series. Notably, we show how to represent an implicit copula when sampling from the transformer decoder model. Moreover, by representing each

[*]Equal contribution [1]ServiceNow Research. Correspondence to: All authors <firstname.lastname@servicenow.com>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

observation as a distinct token with its own timestamp, we can naturally handle irregularly sampled times series as well as series with missing values and unequal sampling frequencies. We also show that an efficient two-dimensional attention scheme lets us scale to hundreds of time steps and series using vanilla attention (Bahdanau et al., 2015).

Contributions:

1. We present *Transformer-Attentional Copulas for Time Series* (TACTiS), a highly flexible transformer-based model for large-scale multivariate probabilistic time series prediction (§4).
2. We introduce *attentional copulas*, an attention-based architecture that estimates non-parametric copulas for an arbitrary number of random variables (§4.2).
3. We theoretically prove the convergence of attentional copulas to valid copulas (§4.3).
4. We conduct an empirical study showing TACTiS' state-of-the-art probabilistic prediction accuracy on several real-world datasets, along with notable flexibility (§5).

2. Background

2.1. Problem Setting

We are interested in the general problem of estimating the joint distribution of values missing at arbitrary time points in multivariate time series. This general task encompasses classical problems, such as probabilistic forecasting, backcasting, and interpolation.

Formally, we consider a set of m multivariate time series $\mathcal{S} \stackrel{\text{def}}{=} \{\mathbf{X}_1, \dots, \mathbf{X}_m\}$, where each $\mathbf{X} \in \mathcal{S}$ is a collection of possibly-related univariate time series. For simplicity, we elaborate the notation for a single element of $\mathcal{S}$. Let $\mathbf{X} \stackrel{\text{def}}{=} \left\{\mathbf{x}_i \in \mathbb{R}^{l_i}\right\}_{i=1}^{n}$, where the $\mathbf{x}_i$ are univariate time series with arbitrary lengths $l_i \in \mathbb{N}^+$. Each $\mathbf{x}_i$ is associated with (*i*) a Boolean mask $\mathbf{m}_i \in \mathbb{B}^{l_i}$, such that $m_{ij} = 1$ if x_{ij} is observed and $m_{ij} = 0$ otherwise, (*ii*) a matrix of time-varying covariates $\mathbf{C}_i \stackrel{\text{def}}{=} [\mathbf{c}_{i1}, \dots, \mathbf{c}_{il_i}] \in \mathbb{R}^{p \times l_i}$, where each $\mathbf{c}_{ij} \in \mathbb{R}^p$ represents arbitrary additional information about the observations, and (*iii*) a vector of time stamps $\mathbf{t}_i \in \mathbb{R}^{l_i}$ s.t. $t_{ij} < t_{i,j+1}$, indicating the times at which the data were measured. Note that this setting naturally supports unaligned time series with arbitrary sampling frequencies.

Our goal is to infer the joint distribution of missing time series values given all known information:[1]

$$P\left(\left\{\mathbf{x}_i^{(m)}\right\}_{i=1}^{n} \;\middle|\; \left\{\mathbf{x}_i^{(o)}, \mathbf{C}_i, \mathbf{t}_i\right\}_{i=1}^{n}\right), \tag{1}$$

where $\mathbf{x}_i^{(o)}$ and $\mathbf{x}_i^{(m)}$ are the observed ($m_{ij} = 1$) and missing ($m_{ij} = 0$) elements of $\mathbf{x}_i$, respectively.

From this problem formulation, one can recover standard time series problems via specific masking patterns. For instance, a t-step probabilistic forecasting task can be defined by setting the last t elements of each $\mathbf{m}_i$ to zero.

2.2. Transformers

The model that we propose builds on the *transformer architecture* for sequence-to-sequence transduction (Vaswani et al., 2017). Transformer models have an encoder-decoder structure, where the encoder learns a representation of the tokens of an input sequence, and the decoder generates the tokens of an output sequence autoregressively, based on the input sequence. The main feature of such models is that they can capture non-sequential dependencies between tokens via attention mechanisms (Bahdanau et al., 2015). This is in sharp contrast with recurrent neural networks (Goodfellow et al., 2016), such as LSTMs (Hochreiter & Schmidhuber, 1997), which are inherently sequential. Transformers have been widely discussed in the literature and we thus refer the reader to the seminal work of Vaswani et al. (2017) for additional details. As we later describe, transformers allow TACTiS to view time series as sets of tokens, among which non-local dependencies can be learned regardless of considerations like alignment and sampling frequency.

2.3. Copulas

Copulas are mathematical constructs that allow separating the joint dependency structure of a set of random variables from their marginal distributions (Nelsen, 2007). Such a separation can be used to learn reusable models that can be applied to seemingly different distributions, where variables have different marginals but share the dependency structure.

Formally, a copula $C : [0, 1]^d \to [0, 1]$ is the joint cumulative distribution function (CDF) of a d-dimensional random vector $[U_1, \dots, U_d]$ on the unit cube with uniform marginal distributions, i.e.,

$$C(u_1, \dots, u_d) \stackrel{\text{def}}{=} P(U_1 \leq u_1, \dots, U_d \leq u_d), \tag{2}$$

where $U_i \sim U_{[0,1]}$. According to Sklar (1959)'s theorem, the joint CDF of any random vector $[X_1, \dots, X_d]$ can be expressed as a combination of a copula C and the marginal CDF of each random variable $F_i(x_i) \stackrel{\text{def}}{=} P(X_i \leq x_i)$,

$$P(X_1 \leq x_1, \dots, X_d \leq x_d) = C\big(F_1(x_1), \dots, F_d(x_d)\big), \tag{3}$$

and the corresponding probability density[2] is given by:

$$\begin{aligned} &p(X_1 = x_1, \dots, X_d = x_d) = \\ &\quad c\big(F_1(x_1), \dots, F_d(x_d)\big) \times f_1(x_1) \times \cdots \times f_d(x_d), \end{aligned} \tag{4}$$

where $c : [0, 1]^d \to [0, 1]$ is the copula's density function and f_i is the marginal density function of X_i. It is thus

[1] We slightly abuse the notation and omit random variables.

[2] Assuming that the distribution is continuous.

possible to estimate seemingly complex joint distributions by learning the parameters of a simple internal joint distribution (the copula) and marginal distributions, which can even be estimated empirically. One classical approach, which has been abundantly used in time series applications, is the Gaussian copula (Nelsen, 2007), where C is constructed from a Gaussian distribution.

In this work, we avoid making such parametric assumptions and propose a new attention-based architecture trained to mimic a non-parametric copula, which we term *attentional copula*. TACTiS learns to produce the parameters of the copula, on-the-fly, based on learned variable representations. It can thus reuse a learned dependency structure across multiple sets of variables, by mapping them to similar representations.

3. Related Work

Neural Networks for Time Series Forecasting Although studied in the 1990's (Zhang et al., 1998), neural networks and other machine learning (ML) techniques had long been taken with caution in the forecasting community due to a perceived propensity to overfit (Makridakis et al., 2018). In recent years, however, the field has seen a number of demonstrations of successful ML-based forecasting, in particular winning the prestigious M5 competition (Makridakis et al., 2021; 2022). For neural networks, work has mostly centered around so-called global models (Montero-Manso & Hyndman, 2021), which in contrast to classical statistical methods such as ARIMA (Box et al., 2015) or exponential smoothing (Hyndman et al., 2008), learn a single set of parameters to forecast many series. A first wave of approaches in the recent resurgence was primarily based on recurrent or convolutional neural network encoders (Shih et al., 2019; Chen et al., 2020). Oreshkin et al. (2020) introduce a recursive decomposition based on a residual signal projection on a set of learned basis functions. Le Guen & Thome (2020) introduce an approach for univariate probabilistic forecasting based on determinantal point processes to capture structured shape and temporal diversity. Extensions of classical state-space models have also been proposed (Yanchenko & Mukherjee, 2020; de Bézenac et al., 2020). Comprehensive surveys of deep learning methods for forecasting appear in Lim & Zohren (2021) and Benidis et al. (2020), restricting coverage to regularly-sampled data. Of relevance to the present work are studies of probabilistic multivariate methods, transformer-based approaches, copulas, and techniques enabling irregular sampling.

Probabilistic Multivariate Methods Narrowing attention to methods carrying out probabilistic forecasting of the joint distribution of a multivariate stochastic process, DeepAR (Salinas et al., 2020) computes an iterated one-step-ahead Monte Carlo approximation of the predictive distribution by sampling from a fixed functional form whose parameters are the result of a recurrent neural network (RNN). Likewise, Rasul et al. (2021a) propose instead to model the predictive joint one-step-ahead distribution using a denoising diffusion process (Ho et al., 2020; Sohl-Dickstein et al., 2015); however, with the diffusion dynamics conditioned on an RNN, the model cannot easily deal with missing values or irregularly sampled time series. Rasul et al. (2021b) propose to model the predictive joint one-step-ahead distribution by a multivariate normalizing flow (Papamakarios et al., 2021), parametrized by either a RNN or a transformer. For general density estimation, Uria et al. (2014) propose an order-agnostic autoregressive density estimator based on neural networks, with Hoogeboom et al. (2022) extending the approach to diffusion models; both are related to the autoregressive copula decomposition that we propose herein.

Transformer-based approaches have come to the fore more recently, on the strength of their successes in other sequence modeling tasks. Lim et al. (2021) introduce the temporal fusion transformer, combining recurrent layers for local processing and self-attention layers for characterizing long-term dependencies, evaluating performance on quantile loss measures; notably, the architecture makes use of a gating mechanism to suppress unnecessary covariates. Li et al. (2019) introduce a transformer with subquadratic memory complexity along with convolutional self-attention to better handle local context. Spadon et al. (2021) propose a recurrent graph evolution neural networks, which embeds transformers, to carry out point multivariate forecasting. Wu et al. (2020) propose an adversarial sparse transformer to estimate conditional quantiles of the predictive distribution. Tashiro et al. (2021) use a conditional score-based diffusion model explicitly trained for interpolation and report substantial improvements over existing probabilistic imputation models, as well as competitive performance on some forecasting tasks against recently-proposed deep learning models. Tang & Matteson (2021) introduce a variational non-Markovian state space model where the latent dynamics are given by an attention mechanism over all previous latents and observed tokens. They report good multivariate probabilistic forecasting results on five standard datasets, as well as on the task of human motion prediction. Wu et al. (2021) introduce an auto-correlation mechanism in place of self attention and report good accuracy on long-horizon point forecasting benchmarks. Recently, outside of time series forecasting, Müller et al. (2022) show how to train a transformer-based model to approximate the Bayesian posterior predictive distribution in regression tasks; these results are in the spirit of the forecasting and interpolation results that we present in this paper.

Copulas-Based Forecasting There exists an abundant literature on uses of copulas (Größer & Okhrin, 2021) for economic and financial forecasting, although most published models have focused on fixed functional forms (Patton, 2012; Rémillard et al., 2012). Aas et al. (2009) introduce vine cop-

ulas that provide increased pairwise flexibility, and Krupskii & Joe (2020); Mayer & Wied (2021) suggest more flexible forms for forecasting applications. In ML, Lopez-Paz et al. (2012) propose to carry out domain adaptation using a form of non-parametric copulas based on kernel estimators. For forecasting, Salinas et al. (2019) introduce GPVar, an LSTM-based method that dynamically parametrizes a Gaussian copula. To the best of our knowledge, none of the published methods show how to sample from a *non-parametric copula* resulting from an autoregressive decomposition of a time-varying conditional copula distribution, as we introduce in this work.

Irregular Sampling Gaussian processes have been widely used to handle irregularly-sampled series (Williams & Rasmussen, 2006; Chapados & Bengio, 2007), but these approaches are limited in other ways, notably in computational tractability. More recently, Shukla & Marlin (2021a) have proposed a transformer-like attention mechanism to re-represent an irregularly sampled time series at a fixed set of reference points, and evaluate on interpolation and classification tasks.

4. The TACTiS Model

Our contribution is a flexible model for multivariate probabilistic time series prediction composed of a transformer encoder (Vaswani et al., 2017) and an attention-based decoder trained to mimic a non-parametric copula; hence the name *Transformer-Attentional Copulas for Time Series* (TACTiS). Fig. 1 shows an overview of the model architecture.[3]

Akin to classical Transformers (Vaswani et al., 2017), TACTiS views the elements of a multivariate time series (x_{ij}) as an arbitrary set of tokens, where some tokens are observed and some are missing (based on m_{ij}). The encoder is tasked with learning a meaningful representation of each token, such as to enable the decoder to infer a multivariate joint distribution over the values of the missing tokens. Due to the use of attention in both the encoder and the decoder, TACTiS can adapt to an arbitrary number of tokens, without retraining.

TACTiS learns the conditional predictive distribution of arbitrary missing values in multivariate time series. At inference time, it is the pattern of missing values themselves—namely, where they are located with respect to measured values—that determines whether it performs forecasting, interpolation, or another similar task. Consequently, TACTiS naturally supports changes in its inputs, such as the unavailability of some time series, and in its outputs, such as changes in forecast horizon. Moreover, it inherently supports misalignment and differences in sampling frequencies, since each token is encoded separately. This is in sharp contrast with classical vector autoregressive models (Wei, 2018) and most recurrent neural networks, which jointly consider the

[3]Code available at https://github.com/servicenow/tactis.

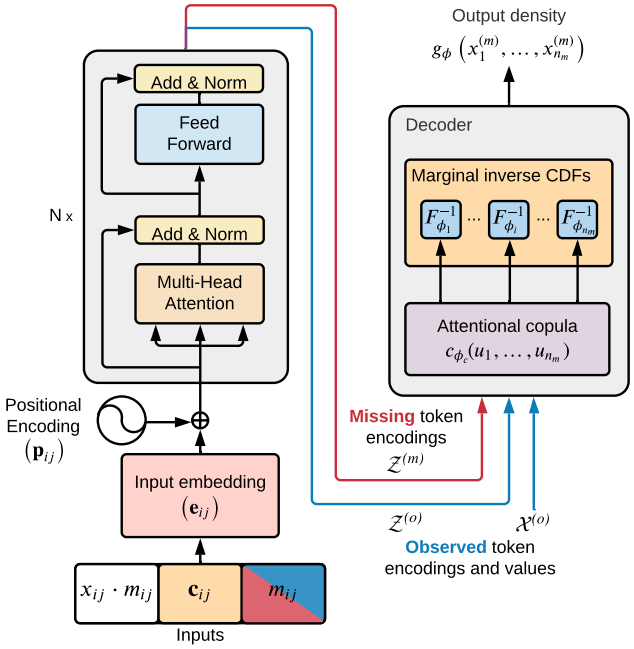


Figure 1. Model overview. **(Left)** The TACTiS encoder is very similar to that of standard transformers. The key difference is that both observed and missing tokens are encoded simultaneously. **(Right)** The decoder, based on an attentional copula, learns the output density given representations of the observed and missing tokens.

values of each time series at a given time step (j) as a vector $(x_{1j}, \ldots, x_{nj}) \in \mathbb{R}^n$. We now detail the encoder, decoder, training procedure, and provide some theoretical guarantees.

4.1. Encoder

As shown in Fig. 1, the encoder used in TACTiS is identical to that of standard Transformers (Vaswani et al., 2017), except that all tokens (observed and missing) are jointly encoded.

Input embedding The encoder starts by producing a vector embedding ($\mathbf{e}_{ij} \in \mathbb{R}^{d_{\text{emb}}}$) for each element in each time series (i.e., tokens), which accounts for its value x_{ij}, the associated covariates[4] $\mathbf{c}_{ij}$, and whether it is observed or missing m_{ij}. Such embeddings are given by a neural network with parameters θ_{emb}, with $x_{ij} \cdot m_{ij}$ masking the values of missing tokens:

$$\mathbf{e}_{ij} = \text{Embed}_{\theta_{\text{emb}}}(x_{ij} \cdot m_{ij}, \mathbf{c}_{ij}, m_{ij}) \,.$$

Positional encoding We add information about a token's time stamp t_{ij} to the input embedding via a positional encoding $\mathbf{p}_{ij} \in \mathbb{R}^{d_{\text{emb}}}$. For simplicity, we use the positional encodings of Vaswani et al. (2017), based on sine and cosine functions of various frequencies, and obtain the final embeddings as $\mathbf{e}'_{ij} = \mathbf{e}_{ij}\sqrt{d_{\text{emb}}} + \mathbf{p}_{ij}$. Note that other choices, such as positional encodings tailored to time series (e.g., accounting for holidays, day of week, etc.), would be

[4]Optionally, we include a learned, per series, embedding in $\mathbf{c}_{ij}$.

viable, but we keep such explorations for future work.

Then, following Vaswani et al. (2017), the $\mathbf{e}'_{ij}$ embeddings are passed through a stack of residual layers that combine multi-head self-attention and layer normalization to obtain an encoding ($\mathbf{z}_{ij}$) for each token. Such encodings contain a complex mixture of information from other tokens that were deemed relevant by the attention mechanism, such as the value of their covariates, their time stamp, and the values of observed tokens.

Scalability One well-known limitation of transformers is the large time and memory complexity of self-attention, scaling quadratically with the number of tokens. However, improving the efficiency of transformers is an active field of research (Tay et al., 2020; Lin et al., 2021) and any progress in this direction is poised to be directly applicable to TACTiS. In this work, when applying TACTiS to large datasets, such as those in § 5, we take advantage of the fact that each token is indexed by two independent indices: i for the variables, and j for the time steps. We do so by employing the temporal transformer layers of Tashiro et al. (2021), which first compute self-attention between the tokens of each variable ($x_{i1}, \ldots, x_{il_i}$) and then between the tokens at a given time step ($x_{1j}, \ldots, x_{nj}$). Hence, instead of scaling in $O([n \cdot l_{\text{max}}]^2)$, the time and memory complexity scale in $O(n^2 \cdot l_{\text{max}} + n \cdot l_{\text{max}}^2)$, where n is the number of time series and $l_{\text{max}} \stackrel{\text{def}}{=} \max_i l_i$ is the length of the longest time series. One downside of this approach is that, for attention within a given time step (j) to make sense, the time series must be aligned. When using temporal transformer layers, we will refer to our model as TACTiS-TT.

4.2. Decoder

As stated in Eq. (1), we aim to learn the joint distribution of the values (x_{ij}) of missing tokens ($m_{ij} = 0$), given the values of observed tokens ($m_{ij} = 1$), the covariates ($\mathbf{c}_{ij}$), and the time stamps (t_{ij}). We achieve this via an attention-based decoder trained to mimic a non-parametric copula (see § 2.3), which we now describe.

Since the data consists of an arbitrary set of observed and missing tokens, we introduce the following notation: $\mathcal{Z}^{(o)} \stackrel{\text{def}}{=} \{\mathbf{z}_1^{(o)}, \ldots, \mathbf{z}_{n_o}^{(o)}\}$, $\mathcal{Z}^{(m)} \stackrel{\text{def}}{=} \{\mathbf{z}_1^{(m)}, \ldots, \mathbf{z}_{n_m}^{(m)}\}$, $\mathcal{X}^{(o)} \stackrel{\text{def}}{=} \{x_1^{(o)}, \ldots, x_{n_o}^{(o)}\}$, and $\mathcal{X}^{(m)} \stackrel{\text{def}}{=} \{x_1^{(m)}, \ldots, x_{n_m}^{(m)}\}$, which respectively denote the encoded representations and values of observed and missing tokens. We also use $\mathcal{C} \stackrel{\text{def}}{=} \{\mathbf{C}_1, \ldots, \mathbf{C}_n\}$ and $\mathcal{T} \stackrel{\text{def}}{=} \{\mathbf{t}_1, \ldots, \mathbf{t}_n\}$ to denote the set of all covariates and timestamps, respectively (see § 2.1).

Our goal is to accurately estimate the joint density of missing token values, using a model $g_\phi(x_1^{(m)}, \ldots, x_{n_m}^{(m)})$ such that:

$$g_\phi(x_1^{(m)}, \ldots, x_{n_m}^{(m)}) \approx p(x_1^{(m)}, \ldots, x_{n_m}^{(m)} \mid \mathcal{X}^{(o)}, \mathcal{C}, \mathcal{T}),$$

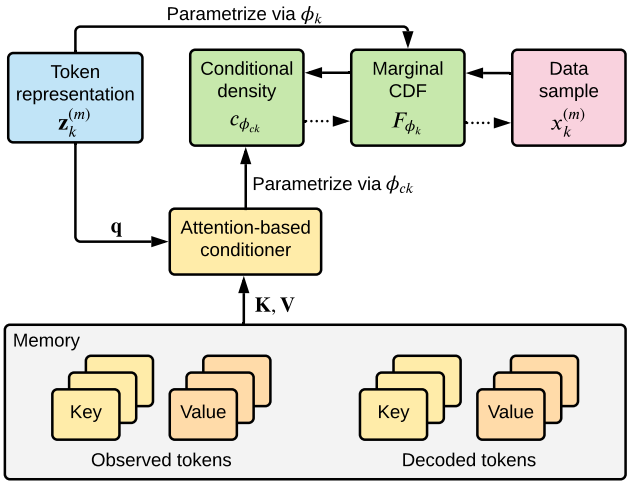


Figure 2. Overview of the TACTiS decoder architecture. Dotted arrows indicate the flow of information during sampling.

where the distributional parameters[5] ϕ are produced by a neural network parametrized by θ_{dec}:

$$\phi = \text{Decoder}_{\theta_{\text{dec}}}(\mathcal{X}^{(o)}, \mathcal{Z}^{(o)}, \mathcal{Z}^{(m)}).$$

Crucially, note that using encoded token representations ($\mathcal{Z}^{(o)}, \mathcal{Z}^{(m)}$; see § 4.1) as inputs into the decoder ensures that the resulting density is conditioned on observations. We consider the following copula-based structure for g_ϕ:

$$g_\phi(x_1^{(m)}, \ldots, x_{n_m}^{(m)}) \stackrel{\text{def}}{=} c_{\phi_c}\left(F_{\phi_1}(x_1^{(m)}), \ldots, F_{\phi_{n_m}}(x_{n_m}^{(m)})\right) \times f_{\phi_1}(x_1^{(m)}) \times \cdots \times f_{\phi_{n_m}}(x_{n_m}^{(m)}), \quad (5)$$

where $\phi \stackrel{\text{def}}{=} \{\phi_c, \phi_1, \ldots, \phi_{n_m}\}$, c_{ϕ_c} is the density of a copula, and F_{ϕ_k} and f_{ϕ_k} respectively represent the marginal (univariate) cumulative distribution and density functions of $X_k^{(m)}$.

The structure of g_ϕ allows for a wealth of different copula parametrizations (e.g., the Gaussian copula of Salinas et al. (2019)). The crux of TACTiS resides in how each of the components is parametrized. Notably, we propose to use normalizing flows (Tabak & Turner, 2013) to model the marginals, and we develop a flexible non-parametric copula that can automatically adapt to a changing number of missing tokens (n_m). We now outline these constructs, for which an overview appears in Fig. 2.

Marginal distributions To model the marginal CDFs (F_{ϕ_k}), we seek a univariate function that (*i*) maps values to $[0, 1]$, (*ii*) is monotonically increasing, (*iii*) is both continuous and differentiable. We achieve this using a

[5] We use subscripted versions of ϕ and θ to denote distributional parameters and neural network parameters, respectively.

modified version of the Deep Sigmoidal Flows (DSF) of Huang et al. (2018), where we simply remove the logit function of the last flow layer to obtain values in $[0, 1]$. The parameters of each flow are produced by a neural network with parameters θ_{F}, which are shared across all k,

$$\phi_k = \text{MarginalParams}_{\theta_{\text{F}}}\left(\mathbf{z}_k^{(m)}\right). \tag{6}$$

The marginal densities f_{ϕ_k} are obtained by differentiating F_{ϕ_k} w.r.t. $x_k^{(m)}$, an efficient operation for DSF. In addition to satisfying our desiderata, DSFs have been shown to be universal density approximators, thereby not constraining the modeling ability of TACTiS.

Copula density According to the definition of a copula (§ 2.3), we must parametrize a distribution on the unit cube $[0, 1]^{n_m}$ with uniform marginals. We consider an autoregressive factorization of the copula density according to an arbitrary permutation $\pi = [\pi_1, \ldots, \pi_{n_m}]$ of the indices $\{1, \ldots, n_m\}$. We denote the copula density and its parameters by $c_{\phi_c^\pi}$:

$$\begin{aligned} c_{\phi_c^\pi}(u_1, \ldots, u_{n_m}) = \\ c_{\phi_{c1}^\pi}(u_{\pi_1}) \times c_{\phi_{c2}^\pi}(u_{\pi_2} \mid u_{\pi_1}) \times \cdots \times \\ c_{\phi_{cn_m}^\pi}\left(u_{\pi_{n_m}} \mid u_{\pi_1}, \ldots, u_{\pi_{n_m-1}}\right), \end{aligned} \tag{7}$$

where $c_{\phi_{ck}^\pi}$ is k^{th} conditional density in the factorization and $u_{\pi_k} = F_{\phi_{\pi_k}}\left(x_{\pi_k}^{(m)}\right)$. Importantly, we let $c_{\phi_{c1}^\pi}$ be the *density of a uniform distribution* $U_{[0,1]}$, and we use an *attention-based conditioner* to obtain the parameters of the remaining conditional distributions ϕ_{ck}^π, for $k > 1$. We call the resulting construct an *attentional copula*.

Attention-based conditioner This component of the decoder produces the parameters ϕ_{ck}^π for the conditional density $c_{\phi_{ck}^\pi}\left(u_{\pi_k} \mid u_{\pi_1}, \ldots, u_{\pi_{k-1}}\right)$ by performing attention over a memory composed of the representations of observed tokens and missing tokens that are predecessors in the permutation: $\left\{\mathbf{z}_1^{(o)}, \ldots, \mathbf{z}_{n_o}^{(o)}, \mathbf{z}_{\pi_1}^{(m)}, \ldots, \mathbf{z}_{\pi_{k-1}}^{(m)}\right\}$, as well as their CDF-transformed[6] values $\left\{u_1^{(o)}, \ldots, u_{n_o}^{(o)}, u_{\pi_1}^{(m)}, \ldots, u_{\pi_{k-1}}^{(m)}\right\}$. The conditioner is composed of several layers, which are as follows. First, we calculate keys $\mathbf{k} \in \mathbb{R}^{d_{\text{att}}}$ and values $\mathbf{v} \in \mathbb{R}^{d_{\text{att}}}$ for each element $(\mathbf{z}, u)$ in the memory using two modules parameterized by θ_{k} and θ_{v}, respectively: [7]

$$\mathbf{k} = \text{Key}_{\theta_{\text{k}}}(\mathbf{z}, u) \qquad \mathbf{v} = \text{Value}_{\theta_{\text{v}}}(\mathbf{z}, u). \tag{8}$$

We then calculate a query $\mathbf{q} \in \mathbb{R}^{d_{\text{att}}}$ for our token of interest $\mathbf{z}_{\pi_k}^{(m)}$ using a module with parameters θ_{q}:

$$\mathbf{q} = \text{Query}_{\theta_{\text{q}}}\left(\mathbf{z}_{\pi_k}^{(m)}\right).$$

Let $\mathbf{K}$ and $\mathbf{V}$ be the matrices of all keys and values for tokens in the memory, respectively. Following Vaswani et al. (2017), we obtain an attention-based representation $\mathbf{z}'' \in \mathbb{R}^{d_{\text{att}}}$:

$$\begin{aligned} \mathbf{z}' &= \text{LayerNorm}\left(\mathbf{V}^\intercal \text{ Softmax}(\mathbf{K}\mathbf{q}) + \mathbf{z}_{\pi_k}^{(m)}\right), \\ \mathbf{z}'' &= \text{LayerNorm}\left(\text{FeedForward}_{\theta_{\text{FF}}}(\mathbf{z}') + \mathbf{z}'\right), \end{aligned}$$

where $\text{FeedForward}_{\theta_{\text{FF}}}$ is a module with parameters θ_{FF}. We repeat this process from Eq. (8) for each layer, with different parameters, and replacing $\mathbf{z}_{\pi_k}^{(m)}$ by the output $\mathbf{z}''$ of the previous layer. Finally, we obtain the parameters of the conditional distribution using a module parameterized by θ_{dist} applied to the $\mathbf{z}''$ of the last layer:

$$\phi_{ck}^\pi = \text{DistParams}_{\theta_{\text{dist}}}(\mathbf{z}'').$$

Choice of distribution Any distribution with support on $[0, 1]$ can be used to model the conditional distributions $c_{\phi_{ck}}^\pi$. We choose to use a piecewise constant distribution, i.e., we divide the support into a number of bins, each parametrized by a probability density that applies to all the points it contains. Such a distribution can approximate complex multimodal distributions on $[0, 1]$ without making parametric assumptions, similarly to van den Oord et al. (2016). The number of bins is a hyperparameter and controls approximation quality. Other valid choices include the Beta distribution and mixtures thereof.

Sampling We first draw a sample from the copula, autoregressively, following an arbitrary permutation π. As per the definition of attentional copulas, the first element of the permutation is always sampled from a $U_{[0,1]}$. Then, we transform each of the sampled values using their corresponding inverse marginal CDF, i.e., $F_{\phi_k}^{-1}(u_k^{(m)})$. This is possible since DSFs are invertible functions (see §B.2 for details).

This concludes the presentation of the decoder. However, one question remains: what guarantees that $c_{\phi_c^\pi}$ will converge to a valid copula? The key is in the training procedure.

4.3. Training Procedure

Let g_{ϕ^π} be the density estimator described in Eq. (5), where the copula is factorized according to permutation π. Let $\Theta \overset{\text{def}}{=} \{\theta_{\text{enc}}, \theta_{\text{dec}}\}$ be the set containing the parameters of all the components of the encoder and the decoder, respectively. We obtain Θ by minimizing the expected negative log-likelihood of our model over permutations drawn uniformly at random[8] from the set of all permutations Π and samples drawn from the set of all time series $\mathcal{S}$:

$$\underset{\Theta}{\text{argmin}} \underset{\substack{\pi \sim \Pi \\ \mathbf{X} \sim \mathcal{S}}}{\mathbb{E}} -\log g_{\phi^\pi}\left(x_1^{(m)}, \ldots, x_{n_m}^{(m)}\right). \tag{9}$$

[6] For observed tokens, transformation via the normalizing flow does not necessarily correspond to a CDF transform, but it ensures that all values are on a similar scale.

[7] For simplicity, we use a single attention head in the presentation, but in practice, we use multiple (see Vaswani et al. (2017)).

[8] Note that we do not differentiate through the sampling of permutations. We simply consider a random permutation at each forward propagation.

Theorem 1. *The attentional copula $c_{\phi_c^\pi}$ embedded in a density estimator g_{ϕ^π}, as shown in Eq. (5), with distributional parameters ϕ^π given by a decoder with parameters $\theta_{dec} \in \Theta$, where Θ minimizes Eq. (9), is a valid copula.*

Proof. The proof, given in § A, relies on the fact that optimizing Eq. (9) leads to permutation invariance which, by the definition of $c_{\phi_{c1}^\pi}$, results in $U_{[0,1]}$ marginal distributions. The resulting copula $c_{\phi_c^\pi}$ is therefore a valid copula. □

Hence, TACTiS provably learns to disentangle the joint dependency structure of random variables from their marginal distributions via flexible non-parametric attentional copulas.

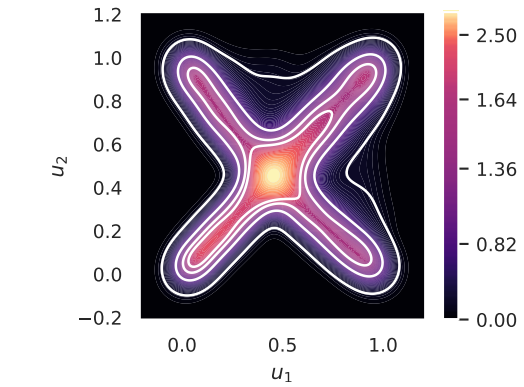
5. Experiments

We start by presenting an experiment that supports the validity of attentional copulas (§5.1). Then, we demonstrate the state-of-the-art performance of TACTiS using a forecasting benchmark composed of several real-world datasets (§5.2). Finally, we present a series of experiments that emphasize the flexibility of the TACTiS model (§5.3).

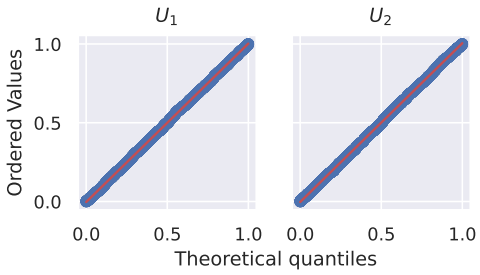
5.1. Empirical Validation of Attentional Copulas

Theorem 1 guarantees that, at convergence to a minimum of Eq. (9), attentional copulas will be valid copulas. However, it does not tell us if this setting is reachable with finite amounts of data, model capacity, and training time. Hence, we conduct a simple experiment, where we generate data from a distribution with a known copula structure and verify if the TACTiS decoder correctly recovers the ground truth. For simplicity and ease of visualization, we use a bivariate distribution, where the underlying copula is an x-shaped mixture of two Clayton copulas. The details are given in §D.1.

The results are shown in Fig. 3. We observe that the learned copula density closely matches the ground truth (Fig. 3a). Furthermore, its marginal distributions are indistinguishable from the $U_{[0,1]}$ distribution (Fig. 3b), making it a valid copula. This experiment, albeit simple, provides empirical evidence that learned attentional copulas can be valid, even in practical settings. See §D.1 for additional results and discussion.



(a) Copula density



(b) Marginal PDFs

Figure 3. Attentional copulas successfully learn valid copulas. a) The learned copula's density (white contours) closely matches the ground truth (heatmap). Note: the support of both distributions is $[0, 1]$; any visible overflow in the figure is due to plotting artefacts. b) The marginal distributions of the learned copula are indistinguishable from $U_{[0,1]}$, as shown by Q-Q plots against $U_{[0,1]}$.

5.2. Forecasting: Comparison to the State of the Art

We now assess the performance of TACTiS in comparison with state-of-the-art forecasting methods. Of particular interest is whether the model's great flexibility is detrimental to the quality of its predictions.

Baselines We benchmark against multiple *deep-learning-based methods* that generate multivariate probabilistic forecasts, namely: GPVar (Salinas et al., 2019), an LSTM-based method that parametrizes a Gaussian copula; TempFlow (Rasul et al., 2021b), a transformer-based method that models the predictive distribution using normalizing flows; and TimeGrad (Rasul et al., 2021a), an autoregressive method based on diffusion models. We also compare to *classical methods*: ARIMA (Box et al., 2015) and ETS exponential smoothing (Hyndman et al., 2008). The comparison is based on TACTiS-TT, a variant of TACTiS that uses temporal transformer layers in the encoder (see §4.1). In addition to these comparisons, we provide a detailed ablation study in §D.4.

Datasets We consider five real-world datasets from the *Monash Time Series Forecasting Repository* (Godahewa et al., 2021): `electricity`, `fred-md`, `kdd-cup`, `solar-10min`, and `traffic` (see §C.1 for details). These were selected for being high-dimensional (107–862 variables), exempt of missing values, and sampled at diverse frequencies (10 min., hourly, monthly).

Evaluation procedure Model accuracy is assessed via a backtesting procedure, which mimics the use of forecasting models in real-world settings. A detailed presentation can be found in §C.5. In short, we define a series of retraining timestamps for each dataset. At each timestamp, the models

Table 1. CRPS-Sum means ($\pm$ standard errors which are autocorrelation-corrected to account for sequential backtesting using the Newey-West (1987; 1994) estimator) for the backtesting benchmark, and average rank of each method across datasets (lower is better for both measures). Best results are in bold.

Model	electricity	fred-md	kdd-cup	solar-10min	traffic	Avg. Rank
Auto-ARIMA	0.077 ± 0.016	0.043 ± 0.005	0.625 ± 0.066	0.994 ± 0.216	0.222 ± 0.005	4.7 ± 0.3
ETS	0.059 ± 0.011	$\mathbf{0.037 \pm 0.010}$	0.408 ± 0.030	0.678 ± 0.097	0.353 ± 0.011	4.4 ± 0.3
TempFlow	0.075 ± 0.024	0.095 ± 0.004	0.250 ± 0.010	0.507 ± 0.034	0.242 ± 0.020	3.9 ± 0.2
TimeGrad	0.067 ± 0.028	0.094 ± 0.030	0.326 ± 0.024	0.540 ± 0.044	0.126 ± 0.019	3.6 ± 0.3
GPVar	0.035 ± 0.011	0.067 ± 0.008	0.290 ± 0.005	$\mathbf{0.254 \pm 0.028}$	0.145 ± 0.010	2.7 ± 0.2
TACTiS-TT	$\mathbf{0.021 \pm 0.005}$	0.042 ± 0.009	$\mathbf{0.237 \pm 0.013}$	0.311 ± 0.061	$\mathbf{0.071 \pm 0.008}$	$\mathbf{1.6 \pm 0.2}$

are trained with all of the preceding data, and their accuracy is assessed using subsequent data. We then report metrics aggregated over all timestamps. The hyperparameters of each method are selected based on the protocol and grids described in §C.3 and §C.4, respectively.

Metrics We use the CRPS-Sum (Salinas et al., 2019), a multivariate extension of the univariate Continuous Ranked Probability Score (CRPS) (Matheson & Winkler, 1976), as our main evaluation metric (see §C.6 for a detailed presentation). In short, this metric corresponds to the CRPS of the univariate series obtained by summing forecasts along the variable axis. For completeness, we also report results for two additional metrics in §C.6: the CRPS and the energy score (Gneiting & Raftery, 2007). Finally, we assess how well each method does as a *general forecasting algorithm*, rather than a dataset-specific one, by measuring the average rank of each method, w.r.t. all others, over all datasets and retraining timestamps.

Benchmark results The CRPS-Sum results are reported in Tab. 1. From these, it is clear that TACTiS-TT compares favourably to the state of the art. It achieves the lowest CRPS-Sum for 3 out of 5 datasets and outperforms most baselines on the remaining ones. In fact, TACTiS-TT outperforms all deep-learning-based methods on fred-md and outperforms all but GPVar on solar-10min. Furthermore, it achieves the lowest average rank (1.6), suggesting that, if one had to choose a method to use without prior knowledge of the data, TACTiS-TT would be the better option. Hence, these results suggest that the great flexibility of TACTiS, which we highlight in the next section, does not seem to undermine its performance.

5.3. Model Flexibility

We now present a series of experiments that emphasize the flexibility of the TACTiS model, namely its support for interpolation, unaligned and non-uniformly sampled data, and its ability to scale to hundreds of time series.

Prediction Beyond Forecasting TACTiS relies on a Boolean-valued mask to determine which values of a multivariate time series must be predicted (see §2.1). This enables it to support arbitrary prediction tasks, such as forecasting, interpolation, and even combinations thereof. Here, we demonstrate support for interpolation by showing that TACTiS can correctly estimate the distribution of a gap in observed values within a stochastic volatility process (Kim et al., 1998). Specifically, we train TACTiS to estimate the distribution of missing values centered within a univariate time series sampled from such a process. We then compare the estimated joint conditional distribution to the ground truth posterior distribution of missing values. A typical result, where TACTiS closely approximates the ground truth, is shown in Fig. 4. Additional results, as well as a full description of the data generation and experimental protocols are available in §D.2.

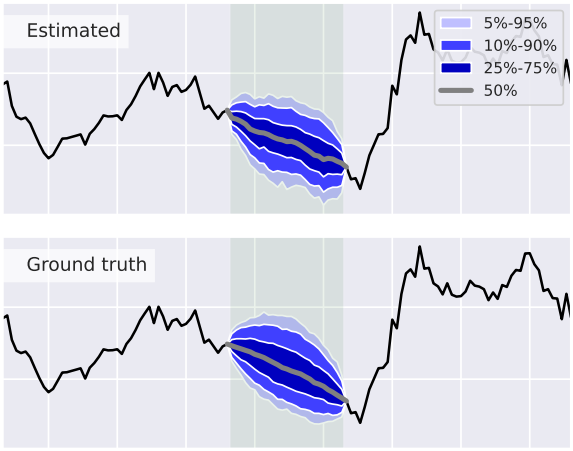


Figure 4. TACTiS *successfully interpolates* missing values (green shaded region) within a stochastic volatility process. The estimated posterior distribution of missing values (top) closely matches the ground truth (bottom).

Unaligned and non-uniformly sampled series One particularity of TACTiS is that it considers each observed data point, in each time series, as a distinct token over which to perform self-attention. The model operates on the set of input tokens, irrespective of their alignment and sampling frequencies, enabling native support for unaligned and non-uniformly sampled time series.[9] Here, we conduct a simple experiment

[9] The TACTiS-TT variant does not support this setting (see §4.1).

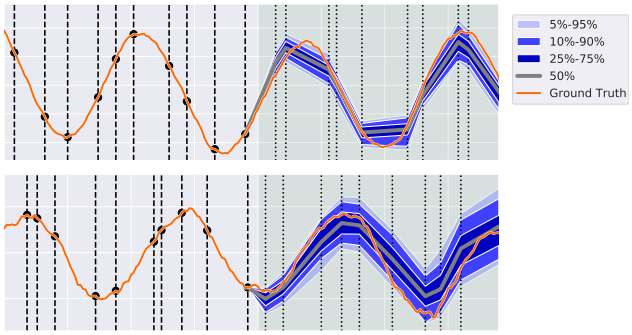


Figure 5. TACTiS *supports unaligned and non-uniformly sampled time series*, as shown in the above forecasts of a bivariate noisy sine process. Observation and prediction timestamps are marked by dashed vertical lines. The forecasted portion of the time series is shaded in green.

Table 2. TACTiS *can be trained efficiently via bagging.* This table shows the mean and standard errors of the CRPS, CRPS-Sum, and energy score metrics for various bagging sizes. The results were obtained using TACTiS-TT on the `fred-md` dataset.

Bagging size	CRPS	CRPS-Sum	Energy
	—	—	$\times 10^5$
2	0.046 ± 0.009	0.039 ± 0.008	7.88 ± 1.70
5	0.052 ± 0.009	0.044 ± 0.008	9.36 ± 1.88
10	0.047 ± 0.009	0.039 ± 0.009	7.90 ± 1.71
15	0.047 ± 0.010	0.040 ± 0.009	7.87 ± 1.78
20	0.049 ± 0.009	0.042 ± 0.008	8.30 ± 1.68
25	0.049 ± 0.010	0.042 ± 0.009	8.22 ± 1.82
30	0.049 ± 0.009	0.043 ± 0.009	8.34 ± 1.74

to demonstrate that TACTiS operates well in this setting. We sample data from a bivariate noisy sine process with observations spaced randomly in each series (see §D.3 for details). We then train TACTiS to forecast the distribution of missing values at the end of each series. As shown in Fig. 5, TACTiS produces accurate forecasts for this data, illustrating its support for unaligned and non-uniformly sampled time series.

Scaling to hundreds of time series One key properties of attention-based models, such as TACTiS, is that they can seamlessly be applied to data of varying dimensionality, without retraining. We make use of this property to devise a scalable training procedure for TACTiS, which we detail in §B.3. In short, we train the model using batches composed of a random subset of $b \ll n$ time series, called a bag ($b = 20$ in the forecasting benchmark). This significantly limits the running time and memory usage of the model. Then, at inference time only, we apply the model to all series. In Tab. 2, we explore the effect of $b \geq 2$ on the predictive performance of TACTiS-TT. The results indicate that this parameter has very little impact on the accuracy of the model. However, as we show in §E.3.1, small values of b, such as $b = 1$, can negatively affect the learned inter-series correlations. Hence, the model can be trained efficiently without incurring a significant penalty in terms of predictive performance as long as the bag size is not unreasonably small.

6. Discussion

This work proposes TACTiS, a method for probabilistic time series inference that combines the flexibility of attention-based models with the density estimation capabilities of a new type of non-parametric copula, termed *attentional copula*. In addition to achieving state-of-the-art performance on tasks such as probabilistic forecasting and interpolation, we showed that TACTiS reaches an unprecedented level of flexibility: it can infer missing values at arbitrary time points in multivariate time series (via masking) can learn from unaligned/non-uniformly sampled data, can be trained when subsets of the data are missing at random, can handle the presence of observed non-stochastic covariates, and can estimate complex distributions beyond the reach of classical copula models, such as the Gaussian copula.

That said, there are several interesting directions in which TACTiS could be extended. First, the model's ability to learn multivariate dependencies may benefit from using positional encodings specifically designed for temporal data, rather than those of Vaswani et al. (2017). Second, the applications of recent advances in large-scale transformers (e.g., Choromanski et al. (2021)) to TACTiS could significantly reduce the amount of resources required by the model, especially in the sampling phase, where bagging is not applied. The elaboration of more efficient sampling procedures (e.g., based on learning conditional independences) also constitutes a promising prospect. Third, a thorough study of the training dynamics of TACTiS may reveal architecture changes or auxiliary tasks that could significantly accelerate learning. Fourth, TACTiS could be extended to series measured in discrete domains by adapting the estimation of marginal distributions in the decoder.

Finally, we believe that this work could serve as a basis for models that address the cold-start problem, making sensible predictions in contexts where very few historical observations of the process are available. In fact, TACTiS could be trained on time series from a wealth of domains, reusing the same attentional copula, but fine-tuning its encoder to new, unforeseen domains. Such extensions towards *foundation models* (Bommasani et al., 2021) for probabilistic time series constitute exciting prospects.

Acknowledgements

The authors are grateful to G. Abuhamad, P. Beaudoin, D. Berger, I. Laradji, C.-W. Huang, A. Lacoste, P.-A. Noël, S. Paquet, and P. Rodriguez-Lopez for thoughtful suggestions.

References

Aas, K., Czado, C., Frigessi, A., and Bakken, H. Pair-copula constructions of multiple dependence. *Insurance: Mathematics and Economics*, 44(2):182–198, 2009. URL https://EconPapers.repec.org/RePEc:eee:insuma:v:44:y:2009:i:2:p:182-198.

Alexandrov, A., Benidis, K., Bohlke-Schneider, M., Flunkert, V., Gasthaus, J., Januschowski, T., Maddix, D. C., Rangapuram, S., Salinas, D., Schulz, J., Stella, L., Türkmen, A. C., and Wang, Y. GluonTS: Probabilistic and Neural Time Series Modeling in Python. *Journal of Machine Learning Research*, 21(116):1–6, 2020. URL http://jmlr.org/papers/v21/19-820.html.

Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. In Bengio, Y. and LeCun, Y. (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1409.0473.

Benidis, K., Rangapuram, S. S., Flunkert, V., Wang, B., Maddix, D., Turkmen, C., Gasthaus, J., Bohlke-Schneider, M., Salinas, D., Stella, L., Callot, L., and Januschowski, T. Neural forecasting: Introduction and literature overview. *arXiv.org*, 2020. URL http://arxiv.org/abs/2004.10240.

Bohlke-Schneider, M. and Salinas, D. personal communication, 2021.

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Box, G. E. P., Jenkins, G. M., Reinsel, G. C., and Ljung, G. M. *Time series analysis: forecasting and control*. John Wiley & Sons, fifth edition, 2015.

Chapados, N. and Bengio, Y. Augmented functional time series representation and forecasting with gaussian processes. In Platt, J., Koller, D., Singer, Y., and Roweis, S. (eds.), *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007. URL https://proceedings.neurips.cc/paper/2007/file/81e74d678581a3bb7a720b019f4f1a93-Paper.pdf.

Chen, Y., Kang, Y., Chen, Y., and Wang, Z. Probabilistic forecasting with temporal convolutional neural network. *Neurocomputing*, 399:491–501, 2020.

Choromanski, K. M., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J. Q., Mohiuddin, A., Kaiser, L., Belanger, D. B., Colwell, L. J., and Weller, A. Rethinking attention with performers. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=Ua6zuk0WRH.

de Bézenac, E., Rangapuram, S. S., Benidis, K., Bohlke-Schneider, M., Kurle, R., Stella, L., Hasson, H., Gallinari, P., and Januschowski, T. Normalizing kalman filters for multivariate time series analysis. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 2995–3007. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/1f47cef5e38c952f94c5d61726027439-Paper.pdf.

Gneiting, T. and Raftery, A. E. Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association*, 102(477):359–378, 2007.

Godahewa, R., Bergmeir, C., Webb, G. I., Hyndman, R. J., and Montero-Manso, P. Monash time series forecasting archive. In *Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021. forthcoming.

Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

Größer, J. and Okhrin, O. Copulae: An overview and recent developments. *WIREs Computational Statistics*, n/a(n/a): e1557, 2021. doi: https://doi.org/10.1002/wics.1557. URL https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/wics.1557. A good initial read to give a higher level of understanding of what copulas are and what kinds of copulas have been studied.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Hoogeboom, E., Gritsenko, A. A., Bastings, J., Poole, B., van den Berg, R., and Salimans, T. Autoregressive diffusion models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=Lm8T39vLDTE.

Huang, C.-W., Krueger, D., Lacoste, A., and Courville, A. Neural autoregressive flows. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp.

2078–2087. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/huang18d.html.

Hyndman, R., Koehler, A. B., Ord, J. K., and Snyder, R. D. *Forecasting with exponential smoothing: the state space approach*. Springer Science & Business Media, 2008.

Hyndman, R., Athanasopoulos, G., Bergmeir, C., Caceres, G., Chhay, L., O'Hara-Wild, M., Petropoulos, F., Razbash, S., Wang, E., and Yasmeen, F. *forecast: Forecasting functions for time series and linear models*, 2022. URL https://pkg.robjhyndman.com/forecast/. R package version 8.16.

Hyndman, R. J. and Khandakar, Y. Automatic time series forecasting: the forecast package for R. *Journal of Statistical Software*, 26(3):1–22, 2008. doi: 10.18637/jss.v027.i03.

Kim, S., Shephard, N., and Chib, S. Stochastic volatility: Likelihood inference and comparison with ARCH models. *The Review of Economic Studies*, 65(3):361–393, 07 1998. ISSN 0034-6527. URL https://doi.org/10.1111/1467-937X.00050.

Koochali, A., Schichtel, P., Dengel, A., and Ahmed, S. Random noise vs state-of-the-art probabilistic forecasting methods: A case study on crps-sum discrimination ability. *arXiv preprint arXiv:2201.08671*, 2022.

Krupskii, P. and Joe, H. Flexible copula models with dynamic dependence and application to financial data. *Econometrics and Statistics*, 16:148–167, 2020. ISSN 2452-3062. doi: https://doi.org/10.1016/j.ecosta.2020.01.005. URL https://www.sciencedirect.com/science/article/pii/S2452306220300216.

Le Guen, V. and Thome, N. Probabilistic time series forecasting with shape and temporal diversity. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 4427–4440. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/2f2b265625d76a6704b08093c652fd79-Paper.pdf.

Li, S., Jin, X., Xuan, Y., Zhou, X., Chen, W., Wang, Y.-X., and Yan, X. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/6775a0635c302542da2c32aa19d86be0-Paper.pdf.

Lim, B. and Zohren, S. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 379(2194):20200209, 2021. doi: 10.1098/rsta.2020.0209. URL https://royalsocietypublishing.org/doi/abs/10.1098/rsta.2020.0209.

Lim, B., Arık, S. Ö., Loeff, N., and Pfister, T. Temporal Fusion Transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4):1748–1764, 2021. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2021.03.012. URL https://www.sciencedirect.com/science/article/pii/S0169207021000637.

Lin, T., Wang, Y., Liu, X., and Qiu, X. A survey of transformers, 2021. URL https://arxiv.org/abs/2106.04554.

Lopez-Paz, D., Hernández-lobato, J., and Schölkopf, B. Semi-supervised domain adaptation with non-parametric copulas. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper/2012/file/8e98d81f8217304975ccb23337bb5761-Paper.pdf.

Makridakis, S., Spiliotis, E., and Assimakopoulos, V. Statistical and machine learning forecasting methods: Concerns and ways forward. *PLoS ONE*, 13(3):e0194889–26, 2018.

Makridakis, S., Spiliotis, E., Assimakopoulos, V., Chen, Z., Gaba, A., Tsetlin, I., and Winkler, R. L. The M5 uncertainty competition: Results, findings and conclusions. *International Journal of Forecasting*, 2021. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2021.10.009. URL https://www.sciencedirect.com/science/article/pii/S0169207021001722.

Makridakis, S., Spiliotis, E., and Assimakopoulos, V. M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting*, 2022. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2021.11.013. URL https://www.sciencedirect.com/science/article/pii/S0169207021001874.

Matheson, J. E. and Winkler, R. L. Scoring rules for continuous probability distributions. *Management science*, 22(10):1087–1096, 1976.

Mayer, A. and Wied, D. Estimation and inference in factor copula models with exogenous covariates, 2021. URL https://arxiv.org/abs/2107.03366.

Montero-Manso, P. and Hyndman, R. J. Principles and algorithms for forecasting groups of time series: Locality and globality. *International Journal of Forecasting*, 37(4):1632–1653, 2021. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2021.03.004. URL https://www.sciencedirect.com/science/article/pii/S0169207021000558.

Müller, S., Hollmann, N., Arango, S. P., Grabocka, J., and Hutter, F. Transformers can do Bayesian inference. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=KSugKcbNf9.

Nelsen, R. B. *An introduction to copulas*. Springer Science & Business Media, second edition, 2007.

Newey, W. K. and West, K. D. A simple, positive semi-definite, heteroskedasticity and autocorrelation consistent covariance matrix. *Econometrica*, 55(3): 703–708, 1987. ISSN 00129682, 14680262. URL http://www.jstor.org/stable/1913610.

Newey, W. K. and West, K. D. Automatic Lag Selection in Covariance Matrix Estimation. *The Review of Economic Studies*, 61(4):631–653, 10 1994. ISSN 0034-6527. doi: 10.2307/2297912. URL https://doi.org/10.2307/2297912.

Oreshkin, B. N., Carpov, D., Chapados, N., and Bengio, Y. 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.

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021. URL http://jmlr.org/papers/v22/19-1028.html.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Patton, A. J. A review of copula models for economic time series. *Journal of Multivariate Analysis*, 110:4–18, 2012.

Peterson, M. *An Introduction to Decision Theory*. Cambridge Introductions to Philosophy. Cambridge University Press, second edition, 2017. doi: 10.1017/9781316585061.

Pinson, P. and Tastu, J. *Discrimination ability of the energy score*. DTU Informatics, 2013.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2020. URL https://www.R-project.org/.

Rasul, K. PyTorchTS, 2021a. URL https://github.com/zalandoresearch/pytorch-ts.

Rasul, K. personal communication, 2021b.

Rasul, K., Seward, C., Schuster, I., and Vollgraf, R. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8857–8868. PMLR, 18–24 Jul 2021a. URL https://proceedings.mlr.press/v139/rasul21a.html.

Rasul, K., Sheikh, A.-S., Schuster, I., Bergmann, U. M., and Vollgraf, R. Multivariate probabilistic time series forecasting via conditioned normalizing flows. In *International Conference on Learning Representations*, 2021b. URL https://openreview.net/forum?id=WiGQBFuVRv.

Rémillard, B., Papageorgiou, N., and Soustra, F. Copula-based semiparametric models for multivariate time series. *Journal of Multivariate Analysis*, 110:30–42, 2012. ISSN 0047-259X. doi: https://doi.org/10.1016/j.jmva.2012.03.001. URL https://www.sciencedirect.com/science/article/pii/S0047259X1200070X. Special Issue on Copula Modeling and Dependence.

Salinas, D., Bohlke-Schneider, M., Callot, L., Medico, R., and Gasthaus, J. High-dimensional multivariate forecasting with low-rank Gaussian copula processes. *Advances in Neural Information Processing Systems*, 32: 6827–6837, 2019.

Salinas, D., Flunkert, V., Gasthaus, J., and Januschowski, T. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3):1181–1191, 2020.

Seabold, S. and Perktold, J. `statsmodels`: Econometric and statistical modeling with Python. In *9th Python in Science Conference*, 2010.

Shih, S.-Y., Sun, F.-K., and Lee, H.-y. Temporal pattern attention for multivariate time series forecasting. *Machine Learning*, 108(8-9):1421–1441, 2019.

Shukla, S. N. and Marlin, B. Multi-time attention networks for irregularly sampled time series. In *International Conference on Learning Representations*, 2021a. URL https://openreview.net/forum?id=4c0J6lwQ4_.

Shukla, S. N. and Marlin, B. M. A survey on principles, models and methods for learning from irregularly sampled time series, 2021b. URL https://arxiv.org/abs/2012.00168.

Sklar, A. Fonctions de répartition à n dimensions et leurs marges. *Publications de l'Institut Statistique de l'Université de Paris*, 8:229–231, 1959.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. In Bach, F. and Blei, D. (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 2256–2265, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/sohl-dickstein15.html.

Spadon, G., Hong, S., Brandoli, B., Matwin, S., Rodrigues-Jr, J. F., and Sun, J. Pay attention to evolution: Time series forecasting with deep graph-evolution learning. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 2021. ISSN 1939-3539. doi: 10.1109/TPAMI.2021.3076155.

Stan Development Team. Stan modeling language users guide and reference manual, 2022. URL https://mc-stan.org.

Sun, C., Hong, S., Song, M., and Li, H. A review of deep learning methods for irregularly sampled medical time series data, 2020. URL https://arxiv.org/abs/2010.12493.

Tabak, E. G. and Turner, C. V. A family of nonparametric density estimation algorithms. *Communications on Pure and Applied Mathematics*, 66(2):145–164, 2013.

Tang, B. and Matteson, D. Probabilistic transformer for time series analysis. *Advances in Neural Information Processing Systems*, 34, 2021.

Tashiro, Y., Song, J., Song, Y., and Ermon, S. CSDI: Conditional score-based diffusion models for probabilistic time series imputation. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Tay, Y., Dehghani, M., Bahri, D., and Metzler, D. Efficient transformers: A survey, 2020. URL https://arxiv.org/abs/2009.06732.

Uria, B., Murray, I., and Larochelle, H. A deep and tractable density estimator. In *International Conference on Machine Learning*, volume 32, pp. 467–475. PMLR, 2014.

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. Wavenet: A generative model for raw audio. In *Arxiv*, 2016. URL https://arxiv.org/abs/1609.03499.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Wei, W. W. *Multivariate time series analysis and applications*. John Wiley & Sons, 2018.

Wiese, M., Knobloch, R., and Korn, R. Copula & marginal flows: Disentangling the marginal from its joint. *arXiv preprint arXiv:1907.03361*, 2019.

Williams, C. K. and Rasmussen, C. E. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

Wu, H., Xu, J., Wang, J., and Long, M. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 22419–22430. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/bcc0d400288793e8bdcd7c19a8ac0c2b-Paper.pdf.

Wu, S., Xiao, X., Ding, Q., Zhao, P., Wei, Y., and Huang, J. Adversarial sparse transformer for time series forecasting. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 17105–17115. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/c6b8c8d762da15fa8dbbdfb6baf9e260-Paper.pdf.

Yanchenko, A. K. and Mukherjee, S. Stanza: A nonlinear state space model for probabilistic inference in non-stationary time series. *arXiv*, pp. 2006.06553v1, 2020.

Zhang, G., Eddy Patuwo, B., and Y. Hu, M. Forecasting with artificial neural networks:: The state of the art. *International Journal of Forecasting*, 14(1):35–62, 1998. ISSN 0169-2070. doi: https://doi.org/10.1016/S0169-2070(97)00044-7. URL https://www.sciencedirect.com/science/article/pii/S0169207097000447.

Appendix – Table of Contents

A. Theory: Proof of Theorem 1

Theorem 1. The copula $c_{\phi_c^\pi}$ embedded in a density estimator g_{ϕ^π}, as shown in Eq. (5), with distributional parameters ϕ^π given by a decoder with parameters $\theta_{\text{dec}} \in \Theta$, where Θ minimizes Eq. (9), is a valid copula.

Proof. To show that $c_{\phi_c^\pi}\left(u_1^{(m)}, \ldots, u_{n_m}^{(m)}\right)$ is the density of a valid copula, we need to show two properties: 1) it is a distribution on the unit cube $[0,1]^{n_m}$, 2) the marginal distribution of each random variable is uniform.

Property (1) is trivially satisfied since, by construction, the support of each conditional distribution is limited to the $[0,1]$ interval (see §4.2, paragraph *Choice of distribution*).

Property (2) is a consequence of minimizing the problem in Eq. (9). Recall that the parameters of the decoder, θ_{dec}, are obtained by minimizing the following expression:

$$\begin{aligned}
\mathop{\mathbb{E}}_{\substack{\pi\sim\Pi \\ \mathbf{X}\sim\mathcal{S}}} -\log g_{\phi^\pi}\left(x_1^{(m)}, \ldots, x_{n_m}^{(m)}\right) &= \mathop{\mathbb{E}}_{\mathbf{X}\sim\mathcal{S}} -\frac{1}{|\Pi|}\sum_{\pi\in\Pi} \log g_{\phi^\pi}\left(x_1^{(m)}, \ldots, x_{n_m}^{(m)}\right) \\
&= \mathop{\mathbb{E}}_{\mathbf{X}\sim\mathcal{S}} -\frac{1}{|\Pi|} \log\left[\prod_{\pi\in\Pi} g_{\phi^\pi}\left(x_1^{(m)}, \ldots, x_{n_m}^{(m)}\right)\right] \\
&= \mathop{\mathbb{E}}_{\mathbf{X}\sim\mathcal{S}} -\log\left[\prod_{\pi\in\Pi} g_{\phi^\pi}\left(x_1^{(m)}, \ldots, x_{n_m}^{(m)}\right)\right]^{|\Pi|^{-1}}. \qquad (10)
\end{aligned}$$

Notice that the term inside the logarithm in Eq. (10) corresponds to a geometric mean. This quantity is always smaller or equal to the arithmetic mean, and equality is reached i.i.f. all elements over which the mean is calculated are equal. Hence, we can rewrite the expression in Eq. (10) as:

$$\mathop{\mathbb{E}}_{\mathbf{X}\sim\mathcal{S}} -\log\left[\frac{1}{|\Pi|}\sum_{\pi\in\Pi} g_{\phi^\pi}\left(x_1^{(m)}, \ldots, x_{n_m}^{(m)}\right)\right] + \delta,$$

where $\delta \in \mathbb{R}^+$ is exactly zero i.i.f. the density estimated by the model is permutation invariant, i.e., $g_{\phi^\pi}\left(x_1^{(m)}, \ldots, x_{n_m}^{(m)}\right) = g_{\phi}\left(x_1^{(m)}, \ldots, x_{n_m}^{(m)}\right), \forall \pi \in \Pi$.

Based on this expression, we conclude that the parameters θ_{dec} that minimize the problem in Eq. (9) lead a density estimator that (i) is invariant to permutations π and (ii) that minimizes the negative log-likelihood of the data $\mathbb{E}_{\mathbf{X}\sim\mathcal{S}} -\log g_{\phi}\left(x_1^{(m)}, \ldots, x_{n_m}^{(m)}\right)$. It naturally follows from Eq. (5) that the embedded copula density $c_{\phi_c^\pi}\left(u_1^{(m)}, \ldots, u_{n_m}^{(m)}\right)$ is also permutation invariant.

Now, recall that, by construction, the marginal density of the first element in a permutation $c_{\phi_{c1}^\pi}(u_1)$, is taken to be that of a $U_{[0,1]}$ (see §4.2, paragraph *Copula density*). Given that the copula density $c_{\phi_{c1}^\pi}\left(u_1^{(m)}, \ldots, u_{n_m}^{(m)}\right)$ is invariant to permutations, the marginal distribution of all variables must necessarily be $U_{[0,1]}$. Thus, *Property (2)* is satisfied.

Since *Properties (1) and (2)* are both satisfied, we conclude that the attentional copula $c_{\phi_c^\pi}$, with parameters obtained from a decoder with parameters θ_{enc} is a valid copula.

□

B. Implementation Details

B.1. Libraries Used

The version of TACTiS used in this work is implemented in PyTorch (Paszke et al., 2019). It relies on the PyTorchTS library (Rasul, 2021a), which allows the integration of PyTorch models with the GluonTS library (Alexandrov et al., 2020), on which we rely heavily in our experiments for data processing, model training, and evaluation. The implementation is available at https://github.com/servicenow/tactis.

B.2. Inverting the Marginal Flows

When sampling from the learned joint distribution, it is needed to compute the inverse of the marginal CDF for each variable, i.e., to invert the marginal flows: $F_{\phi_k}^{-1}(u_k^{(m)})$. Since $F_{\phi_k}(x_k^{(m)})$ is strictly monotonic by construction, many search algorithms can be applied. We choose to rely on binary search due to its numerical stability and ease of implementation. While such searches are relatively slow, the overhead in compute time is negligible compared to other computations in the decoder, which are dominated by the transformer layers in the copula.

One weakness of using flows as marginal distributions is that there is very little pressure toward having well-regularized tails ($u_k^{(m)} \approx 0$ or $u_k^{(m)} \approx 1$). When sampling from these flows, they will rarely produce values that are much smaller or larger than those in the observed data due to these tails not having the correct shape. Given the difficulty of training the flows to avoid this issue entirely, we opted to consider only a portion of the marginals when sampling. That is, instead of sampling from the full $u_k^{(m)} \in [0, 1]$ range in the attentional copula, we rescale sampled values to be in the $[0.05, 0.95]$ range: $x_k^{(m)} = F_{\phi_k}^{-1}(0.05 + 0.9 \times u_k^{(m)})$. This issue and alternative solutions have also been explored in Wiese et al. (2019).

B.3. Bagging: Efficient Training in High Dimensions

Two of the models in our benchmarks: GPVar and TACTiS-TT, can be trained with arbitrary subsets of the n time series in the data without having to adjust their parameters.[10] Hence, the memory footprint of the model during the training phase can be significantly reduced by considering bags of randomly selected time series. In our experiments, each training batch for these models is a bag of 20 time series. At sampling time, the models are applied to the full set of time series.

As mentioned in §C.2.1, we increase the number of batches per epoch to compensate for the reduced amount of data available in each batch due to bagging. See Tab. 2 and §E.3.1 for a study of the effect of the bag size on the accuracy of TACTiS-TT.

B.4. Data Normalization

It is often desirable for a model to be scale and translation invariant. For TACTiS, we thus transform the data according to what we call the "Standardization" procedure. For each sample, we compute the means and variances for each series *using only the values of the observed tokens* $\left\{x_{ij}^{(o)}\right\}$. The values of both the observed and missing tokens is then transformed using

$$\tilde{x}_{ij} = \frac{x_{ij} - \text{mean}_i}{\sqrt{\text{variance}_i}}. \tag{11}$$

After sampling, we can undo this transformation using

$$x_{ij} = \sqrt{\text{variance}_i} \times \tilde{x}_{ij} + \text{mean}_i. \tag{12}$$

Note that we consider a lower bound of 10^{-16} for the variance to avoid division by zero in cases where all values are (nearly) identical. The precise value of this lower bound has no impact on the training when all values are identical. Yet, it has a massive one when sampling since the sampled values $\tilde{x}_{ij}$ are unlikely to all be zero when the marginal flows are not perfectly fitted. Choosing a very small lower bound thus minimizes this issue.

C. Forecasting Benchmark

C.1. Datasets

Tab. 3 describes the five datasets that are included in our benchmark. These datasets were selected due to being publicly available in the Monash Time Series Forecasting Repository (Godahewa et al., 2021), not containing missing values, having a large (but reasonable) number of dimensions, and being sampled at diverse frequencies. A modified version of the `electricity` dataset is often used for benchmarking in the literature, allowing a rough comparison of our results with those of other authors and reinforcing our belief that they are correct. A variant of the `solar-10min` dataset, with hourly frequency, is also often used in the literature. We opted for the 10 minutes version to include such a high-frequency dataset in the benchmark. The prediction length for this dataset was limited to 72 (12 hours) due to a prediction length of 144 (24 hours) being too taxing for the transformer-based models. The `fred-md` dataset was selected due to the nature of its series:

[10] Note that all variants of TACTiS that we consider also support such bagging during training.

Table 3. Detailed presentation of the datasets used in our benchmark, along with the short name by which they are referred to in the paper. Clicking on the short names links to the exact versions of the datasets that were used.

Short name	Monash name	Frequency	Number of series	Prediction length
`electricity`	Electricity Hourly Dataset	1 hour	321	24
`fred-md`	FRED-MD Dataset	1 month	107	12
`kdd-cup`	KDD Cup Dataset (without Missing Values)	1 hour	270	48
`solar-10min`	Solar Dataset (10 Minutes Observations)	10 minutes	137	72
`traffic`	Traffic Hourly Dataset	1 hour	862	24

various economic indicators, which exhibit a wide variety of behaviours. For the `kdd-cup` dataset, a prediction length of 48 (2 days) was used to challenge the models at producing longer forecasts than for the other hourly-frequency datasets. Finally, the `traffic` dataset was selected for being very high-dimensional (862 series).

Each dataset was downloaded from the Monash Repository using GluonTS (Alexandrov et al., 2020) (links to the exact versions are provided in Tab. 3). Apart from some models that perform preprocessing (e.g., standardization), the data were not modified or preprocessed in any way.

C.2. Training Procedure

C.2.1. Deep Learning Models

All deep learning models in our benchmarks: TempFlow, TimeGrad, GPVar, and TACTiS-TT, are trained using the same procedure, except for a few exceptions for GPVar, which we detail below. Each training is done in a Docker container giving access to an NVIDIA Tesla P100 GPU with 12 GB of GPU RAM, 2 CPU cores, and 32 GB of CPU RAM.

Batch size These models are fairly memory-intensive and the amount of resources required varies considerably based on the hyperparameters and the dataset considered. Hence, it was necessary to devise a batch-size selection procedure that would ensure that the model did not outgrow the available resources. To select the batch sizes for a given set of hyperparameters, we ran a small number of training iterations with various batch sizes (powers of two from 1 to 256) and kept the largest one which did not result in an out-of-memory error. While this method is crude, it allows hyperparameters that require less memory to take advantage of the available memory to train faster due to higher parallelization.

Training loop In each training epoch, the models are presented with 1600 random samples from the training set. Since the batch size is variable, we consider $\lfloor 1600/\text{batch size} \rfloor$ batches per epoch. As is explained in §B.3, the GPVar, and TACTiS-TT models use bagging during training and thus only see a random subset of all series at each iteration. To compensate for this, the number of batches per epoch for these models is increased to $\lfloor (1600/\text{batch size}) \times (\text{number of series}/\text{bagging size}) \rfloor$. The batches are built by randomly sampling (uniformly) windows of length equal to the sum of the prediction and history lengths from the training dataset. Only complete windows are considered.

After each epoch, we compute a validation score on a subset of the training set used only for validation (see Fig. 6). The training ends when the first of the following condition is reached:

- The model has trained for 200 epochs,
- The model has trained for more than 3 days, or
- It has been 20 epochs since the best validation score (early stopping).

The resulting model, which is used to compute the final metrics, is the one with minimal validation score. Since we compute the validation score at each epoch, the final model can be from an earlier epoch than the last one performed.

Exceptions for GPVar Following communication with GPVar authors (Bohlke-Schneider & Salinas, 2021), we trained that model on CPU instead of GPU, due to performance concerns. To compensate for not having a GPU, we allocate 8 CPU cores and 64 GB of CPU RAM instead of the 2 CPU cores and 32 GB of CPU RAM allocated to models trained on GPU (see above). Furthermore, we used the GluonTS Trainer for GPVar, which implemented its own stopping conditions for

training. Following what was done for other models, we added a condition which stopped training after a maximum of 3 days.

C.2.2. CLASSICAL MODELS

Our Auto-ARIMA results are obtained by running the `auto.arima` function (Hyndman & Khandakar, 2008) of the `forecast` package (Hyndman et al., 2022) in the R programming language (R Core Team, 2020). This automatically searches the model specification on a per-time series basis. Since the function supports univariate time series only, we run the function independently for each time series (i.e., we treat a d-dimensional multivariate problem as d independent univariate problems), with the hyperparameters and parameters being specific to each time series. We restrict the search to SARIMA models with maximum order $p = 3, q = 3, P = 2, Q = 2$, and the default values for the other function parameters. We carry out an automatic Box-Cox transformation for positive-valued time series. For each time series, we limit fitting time to a maximum of 30 minutes, reverting to a simpler ARMA$(1, 1)$ model with no seasonality or Box-Cox transformation if the original fitting fails. A fit is considered to have failed if the following conditions are encountered:

- Maximum time limit is exceeded;
- The in-sample fitted values contain non-finite values;
- Fewer than 20% of the simulated predictive trajectories contain finite values (e.g. the predictive simulations diverge) or contain values that are not within a factor of 1,000 (in absolute value) of the training observations.

Our ETS (error, trend, seasonality) exponential smoothing (Hyndman et al., 2008) results are obtained through the `ETSModel` implementation within Python's `statsmodels` package (Seabold & Perktold, 2010). This implementation replicates the R implementation within the `forecast` package discussed above. An automatic hyperparameter search is carried out to select, on a *per-dataset basis*, the following hyperparameters:

- Trend: either none or additive;
- Seasonality: either none or additive.

For robustness across a wide variety of datasets, the error term is always considered additive. As with the ARIMA results, since ETS is univariate, independent fitting and predictive simulations of the model are carried out for each time series.

C.3. Hyperparameter Search Protocol

For each model and each dataset under consideration, we ran a hyperparameter search to find the best hyperparameters, which were then used when comparing the forecasting quality of the models. We opted to perform such a search ourselves instead of selecting previously published hyperparameter combinations for two reasons: 1) some datasets that we consider had not been considered when evaluating some methods we compare to, and 2) to maximize the fairness of the comparison by performing an equally extensive search for each model.

In each dataset, the final subset of the time steps is reserved for the backtesting procedure, which will be described in §C.5. From the remaining time steps, a window at the end of length equals to 7 times the prediction length is reserved to serve as the validation set, which is used to compute the metrics used to select the best hyperparameters. All the remaining data, i.e., what comes before the validation set, is used as the training set. See Fig. 6 for a visual representation of this split.

For each model, we considered 50 hyperparameter combinations, randomly selected amongst a range of values defined for each hyperparameter of the model. The models are trained 5 times for each combination of hyperparameters, using random initializations and data sampling orders. For each of these training runs, we compute forecasts using 100 samples for each prediction window in the validation set and average their CRPS-Sum.

We then selected the best hyperparameter combination as being the one where none of the 5 training runs failed due to numerical or memory errors and for which the worst CRPS-Sum value amongst the 5 training runs was the lowest. We considered the worst of the 5 runs instead of the mean or median since we observed that, for some architectures, some hyperparameters led to unstable training, which resulted in significant variability in the quality of the results. The rationale for avoiding such hyperparameter combinations is that such instability would be undesirable in real-world applications.

C.4. Hyperparameter Ranges

In this section, we list the values considered for each hyperparameter of each method. The ranges considered for the baselines are inspired by those published in their respective papers and implementations.

Tab. 4 shows the hyperparameters for the TACTiS-TT model.

Table 4. Possible hyperparameters for TACTiS-TT. $^{\mathbf{e}}$, $^{\mathbf{f}}$, $^{\mathbf{k}}$, $^{\mathbf{s}}$, and $^{\mathbf{t}}$ respectively indicate the optimal hyperparameters for `electricity`, `fred-md`, `kdd-cup`, `solar-10min`, and `traffic`. *The values shown for the Decoder MLP hidden dimensions are the results of our hyperparameters search, not those used in backtesting (see §C.5.1).

	Hyperparameter	Possible values
Model	Encoder transformer embedding size (per head) and feed forward network size	$8^{\mathbf{k}}, 16^{\mathbf{f}}, 24^{\mathbf{est}}$
	Encoder transformer number of heads	$1^{\mathbf{fks}}, 2^{\mathbf{et}}, 3$
	Encoder number of transformer layers pairs	$1^{\mathbf{s}}, 2^{\mathbf{ef}}, 3^{\mathbf{kt}}$
	Encoder time series embedding dimensions	$5^{\mathbf{efkst}}$
	Decoder MLP number of layers	$1^{\mathbf{ekst}}, 2^{\mathbf{f}}, 4$
	Decoder MLP hidden dimensions*	$8^{\mathbf{eft}}, 16, 24^{\mathbf{ks}}$
	Decoder transformer number of heads	$1, 2, 3^{\mathbf{efkst}}$
	Decoder transformer embedding size (per head)	$8^{\mathbf{efst}}, 16^{\mathbf{k}}, 24$
	Decoder number transformer layers	$1^{\mathbf{ef}}, 2^{\mathbf{s}}, 3^{\mathbf{kt}}$
	Decoder number of bins in conditional distribution	$20^{\mathbf{efst}}, 50^{\mathbf{k}}, 100$
	Decoder DSF number of layers	$2^{\mathbf{fks}}, 3^{\mathbf{et}}$
	Decoder DSF hidden dimensions	$8^{\mathbf{ft}}, 16^{\mathbf{eks}}$
	Dropout	$0^{\mathbf{fkst}}, 0.1, 0.01^{\mathbf{e}}$
Data	Normalization	Standardization$^{\mathbf{efkst}}$
	History length to prediction length ratio	$1^{\mathbf{ft}}, 2^{\mathbf{ks}}, 3^{\mathbf{e}}$
Training	Optimizer	RMSprop$^{\mathbf{efkst}}$
	Learning rate	$(10^{-4})^{\mathbf{s}}, (10^{-3})^{\mathbf{efkt}}, (10^{-2})$
	Weight decay	$0^{\mathbf{et}}, (10^{-5})^{\mathbf{ks}}, (10^{-4})^{\mathbf{f}}, (10^{-3})$
	Gradient clipping	$(10^{3})^{\mathbf{efks}}, (10^{4})^{\mathbf{t}}$

Tab. 5 shows the hyperparameters for the TempFlow model. Parameters in `teletype font` refer to parameters in the authors implementation in PyTorchTS (Rasul, 2021a), with all other parameters left at their default value. The choice of possible hyperparameters for the TempFlow model has been discussed with its authors (Rasul, 2021b).

Table 5. Possible hyperparameters for TempFlow. $^{\mathbf{e}}$, $^{\mathbf{f}}$, $^{\mathbf{k}}$, $^{\mathbf{s}}$, and $^{\mathbf{t}}$ respectively indicate the optimal hyperparameters for `electricity`, `fred-md`, `kdd-cup`, `solar-10min`, and `traffic`.

	Hyperparameter	Possible values
Model	`d_model`	$16, 32^{\mathbf{k}}, 64^{\mathbf{es}}, 128^{\mathbf{t}}, 256^{\mathbf{f}}$
	`dim_feedforward_scale`	$1^{\mathbf{e}}, 2, 4^{\mathbf{fkst}}$
	`num_heads`	$1^{\mathbf{f}}, 2^{\mathbf{s}}, 4^{\mathbf{t}}, 8^{\mathbf{ek}}$
	`num_encoder_layers`	$1^{\mathbf{t}}, 3^{\mathbf{k}}, 5^{\mathbf{efs}}$
	`num_decoder_layers`	$1^{\mathbf{et}}, 3^{\mathbf{f}}, 5^{\mathbf{ks}}$
	`dropout_rate`	$0^{\mathbf{est}}, 0.01^{\mathbf{k}}, 0.1^{\mathbf{f}}$
	`flow_type`	`"RealNVP"`$^{\mathbf{ekst}}$, `"MAF"`$^{\mathbf{f}}$
	`n_blocks`	$1^{\mathbf{f}}, 3, 5^{\mathbf{ekst}}$
	`hidden_size`	$8^{\mathbf{t}}, 16^{\mathbf{f}}, 32^{\mathbf{s}}, 64^{\mathbf{k}}, 128^{\mathbf{e}}$
	`n_hidden`	$1, 2^{\mathbf{fk}}, 4^{\mathbf{est}}$
	`conditioning_length`	$8, 16^{\mathbf{fs}}, 32^{\mathbf{t}}, 64^{\mathbf{k}}, 128^{\mathbf{e}}$
	`dequantize`	`False`$^{\mathbf{ekst}}$, `True`$^{\mathbf{f}}$
Data	History length to prediction length ratio	$1^{\mathbf{efk}}, 2^{\mathbf{s}}, 3^{\mathbf{t}}$
Training	Optimizer	Adam$^{\mathbf{efkst}}$
	Learning rate	$(10^{-4})^{\mathbf{fs}}, (10^{-3})^{\mathbf{et}}, (10^{-2})^{\mathbf{k}}$
	Gradient clipping	$(10^{1})^{\mathbf{kst}}, (10^{2})^{\mathbf{ef}}, (10^{4})$

Tab. 6 shows the hyperparameters for the TimeGrad model. Parameters in `teletype font` refer to parameters in the authors implementation in PyTorchTS (Rasul, 2021a), with all other parameters left at their default value.

Table 6. Possible hyperparameters for TimeGrad. $^{\mathbf{e}}$, $^{\mathbf{f}}$, $^{\mathbf{k}}$, $^{\mathbf{s}}$, and $^{\mathbf{t}}$ respectively indicate the optimal hyperparameters for `electricity`, `fred-md`, `kdd-cup`, `solar-10min`, and `traffic`.

	Hyperparameter	Possible values
Model	`num_layers`	$1, 2, 3^{\mathbf{efkst}}$
	`num_cells`	$20^{\mathbf{fkt}}, 40^{\mathbf{s}}, 60^{\mathbf{e}}$
	`dropout_rate`	$0^{\mathbf{fs}}, 0.01^{\mathbf{e}}, 0.1^{\mathbf{kt}}$
	`diff_steps`	$25, 50, 100^{\mathbf{efkst}}$
	`beta_schedule`	$\texttt{"linear"}^{\mathbf{fk}}, \texttt{"quad"}^{\mathbf{est}}$
	`residual_layers`	$4^{\mathbf{e}}, 8^{\mathbf{t}}, 16^{\mathbf{fks}}$
	`residual_channels`	$4^{\mathbf{fks}}, 8, 16^{\mathbf{et}}$
	`scaling`	$\texttt{False}^{\mathbf{t}}, \texttt{True}^{\mathbf{efks}}$
Data	History length to prediction length ratio	$1^{\mathbf{ft}}, 2^{\mathbf{ks}}, 3^{\mathbf{e}}$
Training	Optimizer	$\text{Adam}^{\mathbf{efkst}}$
	Learning rate	$(10^{-4})^{\mathbf{ft}}, (10^{-3})^{\mathbf{e}}, (10^{-2})^{\mathbf{ks}}$
	Gradient clipping	$(10^{1}), (10^{2})^{\mathbf{fks}}, (10^{4})^{\mathbf{et}}$

Tab. 7 shows the hyperparameters for the GPVar model. Parameters in `teletype font` refer to parameters in the authors implementation in GluonTS (Alexandrov et al., 2020), with all other parameters left at their default value.

Table 7. Possible hyperparameters for GPVar. $^{\mathbf{e}}$, $^{\mathbf{f}}$, $^{\mathbf{k}}$, $^{\mathbf{s}}$, and $^{\mathbf{t}}$ respectively indicate the optimal hyperparameters for `electricity`, `fred-md`, `kdd-cup`, `solar-10min`, and `traffic`.

	Hyperparameter	Possible values
Model	`num_layers`	$1^{\mathbf{e}}, 2^{\mathbf{k}}, 4^{\mathbf{fst}}$
	`num_cells`	$8^{\mathbf{t}}, 16^{\mathbf{fs}}, 24^{\mathbf{k}}, 32, 40^{\mathbf{e}}$
	`cell_type`	$\texttt{"lstm"}^{\mathbf{fks}}, \texttt{"gru"}^{\mathbf{et}}$
	`use_marginal_transformation`	$\texttt{True}^{\mathbf{efkst}}, \texttt{False}$
	`rank`	$8^{\mathbf{f}}, 16^{\mathbf{ek}}, 24^{\mathbf{st}}$
	`dropout_rate`	$0^{\mathbf{ft}}, 0.01^{\mathbf{e}}, 0.1^{\mathbf{ks}}$
Data	History length to prediction length ratio	$1, 2^{\mathbf{st}}, 3^{\mathbf{efk}}$
Training	Optimizer	$\text{Adam}^{\mathbf{efkst}}$
	Learning rate	$(10^{-4})^{\mathbf{eft}}, (10^{-3})^{\mathbf{ks}}, (10^{-2})$
	Weight decay	$0^{\mathbf{k}}, (10^{-8})^{\mathbf{ft}}, (10^{-5}), (10^{-4}), (10^{-3})^{\mathbf{es}}$
	Gradient clipping	$10^{\mathbf{efkst}}$

C.5. Backtesting Protocol

We evaluate the performance of all models using a backtesting procedure that mimics how they would be applied to real-world problems. In practice, due to the high cost of hyperparameter search, it would be done seldomly. Model training with fixed hyperparameters is less expensive and would be done periodically. Finally, forecasting using a pre-trained model is cheap and would be done as needed. Therefore, in our framework, we conduct a single hyperparameter search, retrain the model multiple times (periodically), and calculate forecasts at multiple time stamps between retrainings.

For each dataset, we define n_B backtesting timestamps $\tau_1, \tau_2, \ldots, \tau_{n_B}$. For each backtesting timestamps τ_i, we further define multiple forecasting timestamps $\tau_{i1}, \tau_{i2}, \ldots, \tau_{in_F}$. The hyperparameter search described in §C.3 is done using only data before τ_1, while the i-th model is trained using only the data before τ_i. A visual representation of the split between the data used for model training, the historical data used for the forecast, and the target data to which forecasts are compared is shown in Fig. 6.

We use $n_B = 6$ for all datasets. For `electricity`, we select Mondays at midnight as training time τ_i, with one week between

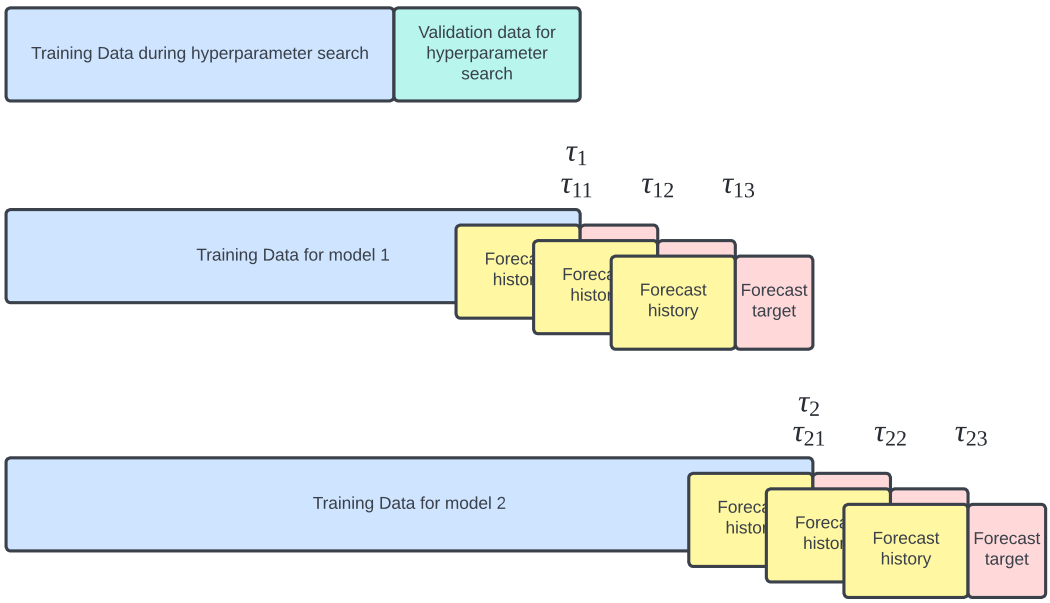


Figure 6. A visual representation of how the data is split during hyperparameter search and backtesting with $n_B = 2$ and $n_F = 3$. During backtesting, the models are trained at times τ_i, and are used for forecasting at times τ_{ij} (with $\tau_i = \tau_{i1}$).

each, and estimate forecasts every 24 hours between them. For `fred-md`, we select Januaries for the τ_i, with one year between each, and estimate only a single forecast at the same time. For `kdd-cup`, we also select Mondays at midnight for the τ_i, with two weeks between each, and estimate forecasts every 48 hours between them. For `solar-10min`, we also select Mondays at midnight for the τ_i, with one week between each, and estimate forecasts every 12 hours between them. For `traffic`, we also select Mondays at midnight for the τ_i, with one week between each, and estimate forecasts every 24 hours between them. The last backtesting timestamp τ_6 for each dataset is selected as the last possible timestamp that follows the respective criterion while still having enough data after it for the full prediction range.

The training is done as explained in §C.2.1, except that we do not perform early stopping using a validation set. Instead, we use the maximum number of epochs that were performed using the best-performing hyperparameters in the hyperparameter search.

C.5.1. Exceptions for TACTiS-TT

In experiments conducted using toy datasets, such as small subsets of our main datasets, we found that TACTiS-TT had complex learning dynamics, which can be broken down into three stages. At first, it quickly improves the quality of the marginal distributions while pushing the attentional copula towards the trivial copula (uniform distribution on the unit cube), where all values are independent. Then, the loss tends to plateau, with no improvement in forecast quality for multiple epochs. Finally, it moves on to learning a non-trivial attentional copula, and its loss improves again.

While these three phases are not as clear-cut in the main datasets as in the toy datasets, the plateau effect in the second learning phase could cause early stopping to be too aggressive and prevent TACTiS-TT from learning a non-trivial copula. Therefore, when training TACTiS-TT in the backtesting procedure, we removed the maximum number of epochs, keeping only the 3 day maximum training duration. In retrospect, this had minimal impact on the number of epochs performed for all datasets, except for the smallest: `fred-md`, for which training reached a number of epochs in the order of 7000 during the 3 days limit.

Another important observation that was made using toy datasets is that too small values for the *MLP hidden dimensions* parameter in the decoder would greatly slow the transition from the second to third phases of training. We thus increased this parameter to 48 for all datasets, ignoring this part of our hyperparameter search.

Given these observations, we believe there is sill work to do in understanding and improving the learning dynamics of TACTiS. Progress in these directions may lead to significant improvement in the model's empirical performance and the quality of

the learned copulas.

C.6. Metrics and Additional Results

As mentioned in §5, we use the CRPS-Sum as our primary metric to select the best hyperparameters and compare the forecasting accuracy of the various models. The CRPS-Sum is based on the Continuous Ranked Probability Score (CRPS) (Matheson & Winkler, 1976), which can be computed for each individual forecasted value for the i-th series and the j-th timestep as:

$$\mathrm{CRPS}(X_{ij}, x_{ij}) = \mathop{\mathbb{E}}_{X_{ij}} \left[|X_{ij} - x_{ij}| \right] - \frac{1}{2} \mathop{\mathbb{E}}_{X_{ij}, X'_{ij}} \left[|X_{ij} - X'_{ij}| \right], \tag{13}$$

where x_{ij} is the observed value, and X_{ij} and X'_{ij} are two independent variables from the forecasted sampling process. The CRPS-Sum is obtained by replacing the individual values by the sum over all series i: $s_j = \sum_i x_{ij}$ and $S_j = \sum_i X_{ij}$. This transformation of the CRPS-Sum allows it to capture the quality of some of the correlations in the forecasts: namely, the average correlation between forecasts at a single time step. However, the CRPS-Sum metric is inherently univariate and cannot discriminate whether the forecasting method accurately predicts the correlation between different time steps or not.

Computing Standard Errors In all our results tables, we compute the standard errors (indicated by the $\pm$) of all metrics using an adjustment to the variance across individual results to account for the autocorrelation that arises from the sequential nature of our backtesting procedure. Specifically, we use a Newey-West correction for standard errors, computed using Bartlett kernel weights (Newey & West, 1987; 1994), with 3 lags (since there were only 6 backtesting folds). The automatic bandwidth selection procedure described in Newey & West (1994) is used, as implemented in the R `sandwich` package. This correction produces more conservative (i.e. wider) standard errors than those computed under an i.i.d. assumption and, as such, yields a fairer assessment of the methods under study.

CRPS Results The benchmark results for the CRPS-Sum and CRPS are respectively shown in Tabs. 1 and 8. We used the GluonTS (Alexandrov et al., 2020) implementation of said metrics, which are thus normalized by dividing the CRPS of each individual variable by the mean absolute value of its observed values.

Table 8. CRPS means ($\pm$ standard errors which are autocorrelation-corrected to account for sequential backtesting using the Newey-West (1987; 1994) estimator) for the backtesting benchmark, and average rank of each method across datasets (lower is better for both measures). Best results are in bold.

Model	`electricity`	`fred-md`	`kdd-cup`	`solar-10min`	`traffic`	**Avg. Rank**
Auto-ARIMA	0.129 ± 0.015	0.052 ± 0.005	0.477 ± 0.015	0.636 ± 0.060	0.310 ± 0.004	4.1 ± 0.3
ETS	0.094 ± 0.014	0.050 ± 0.011	0.560 ± 0.028	0.844 ± 0.119	0.437 ± 0.012	4.9 ± 0.2
TempFlow	0.109 ± 0.024	0.110 ± 0.003	0.451 ± 0.005	0.547 ± 0.036	0.320 ± 0.015	4.1 ± 0.3
TimeGrad	0.101 ± 0.027	0.142 ± 0.058	0.495 ± 0.023	0.560 ± 0.047	0.217 ± 0.015	3.7 ± 0.3
GPVar	0.067 ± 0.010	0.086 ± 0.009	0.459 ± 0.009	$\mathbf{0.298 \pm 0.034}$	0.213 ± 0.009	2.9 ± 0.1
TACTiS-TT	$\mathbf{0.052 \pm 0.006}$	$\mathbf{0.048 \pm 0.010}$	$\mathbf{0.420 \pm 0.007}$	0.326 ± 0.049	$\mathbf{0.161 \pm 0.009}$	$\mathbf{1.4 \pm 0.1}$

Energy Score An alternative metric is the energy score (Gneiting & Raftery, 2007). Like the CRPS and CRPS-Sum metrics, it is a proper scoring method; thus, a perfect forecast would minimize it. Unlike the CRPS and CRPS-Sum metrics, it is sensitive to the correlations between all forecasted variables, thus allowing a good multivariate forecasting technique to shine compared to one that forecasts each series independently. The energy score is defined as:

$$\mathrm{energy}(\{X, x\}) = \mathop{\mathbb{E}}_{X} \|X - x\|_F^\beta - \frac{1}{2} \mathop{\mathbb{E}}_{X, X'} \|X - X'\|_F^\beta, \tag{14}$$

where x is the observed value in matrix form, X and X' are two independent variables from the forecasted sampling process in matrix forms, β is a parameter set to 1, and $\| \cdot \|_F$ is the Frobenius matrix norm.

The benchmark results for the energy score are shown in Tab. 9. Unlike the CRPS, which is normalized, the energy scores we report are not, which explains the orders of magnitude differences between the datasets. However, it should be noted that these results were obtained from models with hyperparameters selected to minimize the CRPS-Sum and not the energy score. It is thus likely that some hyperparameters are not optimal in terms of energy score, so these results may not be the best that the models can do.

Table 9. Energy score means ($\pm$ standard errors which are autocorrelation-corrected to account for sequential backtesting using the Newey-West (1987; 1994) estimator) for the backtesting benchmark, and average rank of each method across datasets (lower is better for both measures). Best results are in bold.

Model	`electricity` $\times 10^4$	`fred-md` $\times 10^5$	`kdd-cup` $\times 10^3$	`solar-10min` $\times 10^2$	`traffic` $\times 10^0$	**Avg. Rank** —
Auto-ARIMA	44.59 ± 8.56	8.72 ± 0.81	18.76 ± 3.31	19.42 ± 3.37	4.10 ± 0.05	4.9 ± 0.2
ETS	7.94 ± 0.93	$\mathbf{7.90 \pm 1.88}$	3.60 ± 0.24	4.74 ± 0.17	4.98 ± 0.07	4.1 ± 0.2
TempFlow	10.25 ± 2.03	20.16 ± 0.74	3.30 ± 0.28	4.25 ± 0.16	4.59 ± 0.25	4.3 ± 0.4
TimeGrad	9.69 ± 2.62	19.87 ± 7.23	3.30 ± 0.19	4.31 ± 0.23	3.38 ± 0.11	3.2 ± 0.4
GPVar	6.80 ± 0.62	11.43 ± 1.60	3.18 ± 0.20	$\mathbf{2.60 \pm 0.10}$	3.57 ± 0.10	2.6 ± 0.2
TACTiS-TT	$\mathbf{5.42 \pm 0.57}$	8.18 ± 1.83	$\mathbf{2.93 \pm 0.22}$	2.88 ± 0.23	$\mathbf{3.10 \pm 0.13}$	$\mathbf{1.8 \pm 0.4}$

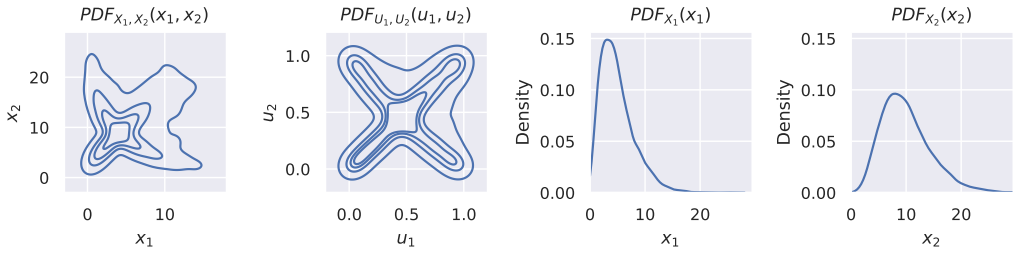


Figure 7. Overview of the simulated dataset: a) contour plot of the joint PDF of X_1 and X_2, b) contour plot of the ground truth copula's PDF, c-d) marginal PDF of X_1 and X_2, respectively. All plots are based on 5000 samples.

D. Additional Experiments

D.1. Can Attentional Copulas Recover a Ground-Truth Copula?

In this experiment, we evaluate the density estimation abilities of the TACTiS decoder. That is, we evaluate its ability to correctly recover the copula and the marginal distributions that underlie a joint distribution. Doesn't the result in Theorem 1 already guarantee this? No, Theorem 1 guarantees that, at convergence to a minimum of Eq. (9), the resulting attentional copula ($c_{\phi_c^\pi}$) will be a valid copula, but it does not tell us if this setting is reachable given finite amounts of data, capacity, and training time. Below, we show that a valid copula can be learned in practical conditions.

Experimental setup For simplicity and ease of visualization, we focus on a simple bivariate joint density estimation problem. We define a bivariate data distribution with known marginal distributions and copula structure. We then train the TACTiS decoder using samples from this distribution and evaluate its ability to recover the ground truth marginals and copula.

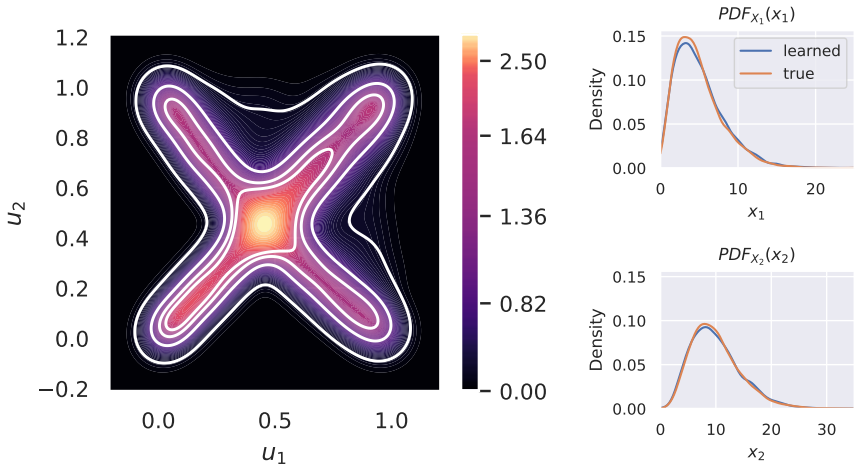
Data generation We define the joint dependency structure of the variables using a uniformly weighted mixture of two Clayton copulas with parameter θ equals to 14.75 and -0.85, respectively. This leads to a complex x-shaped copula density (Fig. 7b). As for the marginal distributions, we use Chi-squared distributions and let $X_1 \sim \chi^2(5)$ and $X_2 \sim \chi^2(10)$, respectively (Fig. 7cd).

Model We isolate the density estimation components of TACTiS, i.e., those tasked with learning the marginal distributions and the copula. As such, we replace the encoder with a simple set of embeddings (one for each variable in the distribution) and use these embeddings as the $\mathcal{Z}^{(m)}$ input to the TACTiS decoder (see §4.2). Since we focus on an unconditional density estimation task, we let the $\mathcal{Z}^{(o)}$ input to the decoder be the empty set. We then train the model by maximum likelihood, using the permutation-based loss in Eq. (9). The full configuration of the model is given in Tab. 10.

Results Our desiderata is that 1) the learned copula and marginal densities closely match those shown in Fig. 7b and Fig. 7cd, respectively and 2) that the learned copula be valid (see §2.3), i.e., a distribution on the unit cube with $U_{[0,1]}$ marginals. As shown in Fig. 8, the learned copula density (a) and the marginal distributions (b) closely approximate the ground truth. Further, as shown in Fig. 9, the marginal distributions of the copula are indistinguishable from the uniform distribution. Hence, we conclude that 1) the TACTiS decoder successfully recovered the components of the data distribution and 2) that the attentional copula successfully converged to a valid copula, and this, even in a setting where data, capacity, and training time were limited.

Table 10. Hyperparameters for TACTiS for the experiments in §D.1.

	Hyperparameter	Selected values
Model	Encoder variable embedding dimensions	3
	Decoder MLP number of layers	2
	Decoder MLP hidden dimensions	30
	Decoder transformer number of heads	1
	Decoder transformer embedding size (per head)	8
	Decoder number transformer layers	2
	Decoder number of bins in conditional distribution	30
	Decoder DSF number of layers	2
	Decoder DSF hidden dimensions	8
	Dropout	0
Training	Optimizer	RMSprop
	Learning rate	(10^{-3})
	Batch size	128
	Weight decay	0
	Gradient clipping	none



(a) Copula density **(b)** Marginal PDFs

Figure 8. The TACTiS decoder successfully recovers the ground truth components of the distribution: a) the learned copula density (white contours) closely matches the ground truth density (heatmap); b) the learned marginals (blue) closely match the ground truth (orange). All plots are based on 5000 samples.

Additional remarks While conducting these experiments, which allow for easy visualization, we realized that initialization seems to play an important role in the model. In fact, at initialization, the bins in the categorical distribution used in the copula have uniform weights, resulting in a trivial, but valid, copula with $U_{[0,1]}$ marginals. While this does not correctly capture the dependency between the variables, it enables the model to concentrate on fitting the marginals very precisely. As such, the model sometimes gets stuck in this regime and fails to learn a non-trivial copula. In these experiments, we circumvented this issue by conducting each experiment with various initializations (via the random seed). As for our main results, the forecasting benchmark experiments, we ensured that this issue did not arise by verifying that a non-zero correlation existed between some variables in the learned distribution (see §D.4). Nonetheless, a thorough study of the learning dynamics of TACTiS is an interesting future direction, which ultimately might help to prevent this pitfall.

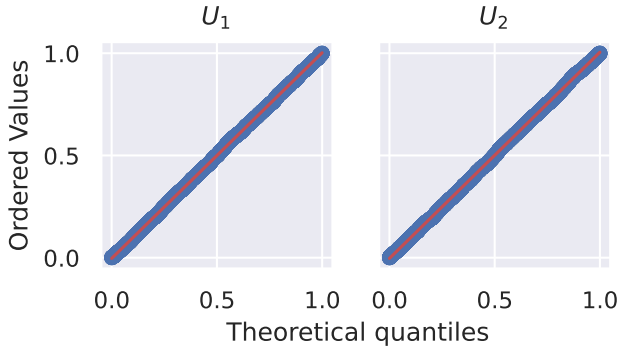


Figure 9. The empirical marginal densities of the learned copula are indistinguishable from that of the $U_{[0,1]}$ distribution, as shown by these Q-Q plots w.r.t. $U_{[0,1]}$. All plots are based on 5000 samples.

D.2. Can TACTiS Learn to Interpolate?

In this section, we evaluate the ability of TACTiS to perform interpolation, i.e., estimating the distribution of missing values for a gap *within* a time series.

Data generation In this experiment, we consider a univariate[11] time series generated according to the following stochastic volatility process (Kim et al., 1998):

$$y_t \mid h_t \sim \mathcal{N}(0, \exp h_t) \tag{15}$$

$$h_t \mid h_{t-1}, \mu, \phi, \sigma \sim \mathcal{N}\left(\mu + \phi(h_{t-1} - \mu), \sigma^2\right) \tag{16}$$

$$h_0 \mid \mu, \phi, \sigma \sim \mathcal{N}\left(\mu, \sigma^2/(1-\phi^2)\right), \tag{17}$$

with a log-variance of level $\mu = -9$, persistence $\phi = 0.99$, and volatility $\sigma = 0.04$. The h_t variables are unobserved and correspond to a latent time-varying volatility process. Our quantity of interest is $x_t \stackrel{\text{def}}{=} x_{t-1} + y_t$, where y_t is a stochastic increment incurred at each time step.

Training data. We use this process to generate a univariate time series $\mathbf{x}^{\text{train}}$ of length $10,000$, with $x_1^{\text{train}} = 1$. This time series is used to train TACTiS.

Evaluation data. We then generate 1000 more univariate time series, denoted $\mathbf{x}_1^{\text{test}}, \ldots \mathbf{x}_{1000}^{\text{test}}$, which correspond to possible trajectories of length 500 starting at the end of $\mathbf{x}^{\text{train}}$. For each $\mathbf{x}_i^{\text{test}}$, we create a gap of missing values of length 25 at an arbitrary time point. We then use the Markov Chain Monte-Carlo sampling capabilities of the Stan probabilistic programming language (Stan Development Team, 2022) to sample possible interpolation trajectories from the posterior distribution of missing values. Such trajectories form a ground truth for the interpolation task associated with $\mathbf{x}_i^{\text{test}}$.

Training protocol We train TACTiS using batches of 64 windows of length 125 drawn randomly from the training sequence $\mathbf{x}^{\text{train}}$. For each window, we mask the values of the center 25 time points and let the other time points (50 on each side) be observed. The model then learns to estimate the distribution of the 25 masked values given the 100 observed ones. The hyperparameter values used in this experiment are reported in Tab. 11.

Evaluation protocol For each of the evaluation time series $\mathbf{x}_i^{\text{test}}$, we extract a region of length 125 centered on the gap of missing values.[12] We then use TACTiS to draw samples from the estimated distribution of missing values. Then, the energy score (see Eq. (14)) is used to compare the sampled trajectories to *each* of the ground truth trajectories for $\mathbf{x}_i^{\text{test}}$. This results in a distribution of energy scores.

For comparison, we repeat the same process and obtain a distribution of energy scores for the following baselines:

- Oracle: an oracle model that produces samples by randomly drawing from the ground truth trajectories for $\mathbf{x}_i^{\text{test}}$.
- Dummy: a simple baseline that performs a linear interpolation between the points directly before and after the gap of

[11] We could also have considered a multivariate setting, but we limit the experiment to a univariate one for simplicity.

[12] We keep only the testing tasks where such sampling was within the bounds of the time series, resulting in 803 valid tasks $\mathbf{x}_i^{\text{test}}$.

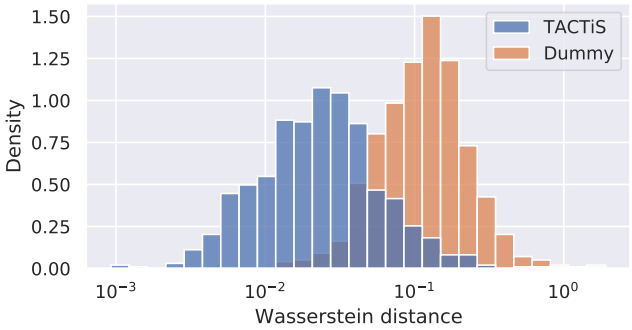


Figure 10. TACTiS outperforms the Dummy baseline on the considered interpolation tasks, as shown by the distribution of Wasserstein distances between the energy score distributions of each baseline and the Oracle (estimated for our 803 valid testing tasks). The TACTiS and Dummy distributions have means at 0.0391 and 0.1459, respectively. The distances are estimated using 50 samples from each energy score distribution.

missing values. This baseline is deterministic, i.e., all its samples are identical.

We then use the Wasserstein distance to compare the distribution of scores obtained for TACTiS and for the Dummy baseline to that of the Oracle, for each testing series $\mathbf{x}_i^{\text{test}}$.

Results and conclusions

Fig. 10 summarizes the Wasserstein distances obtained for each of the testing series. Clearly, the distributions estimated by TACTiS are closer to the ground truth distributions than those estimated by the Dummy baseline. This is reflected in their respective Wasserstein distance distributions, which have a mean of 0.0391 and 0.1459 for TACTiS and the Dummy baseline, respectively. We further assess the quality of the distributions estimated by TACTiS visually. Fig. 11 compares the distribution estimated by TACTiS for the task ($\mathbf{x}_i^{\text{test}}$) where the energy score distribution was closest to that of the Oracle (WD $= 0.0009$). We observe that the shape of the estimated distribution is very close to that of the ground truth. Furthermore, its median is coherent with the values observed directly before and after the gap. We repeat this analysis for the task where the WD was closest to the mean of the TACTiS WD distribution (WD $= 0.0388$) and show the results in Fig. 12. The same conclusions apply, supporting the fact that TACTiS generally performs well at this interpolation task. Finally, we repeat the analysis for the task where the WD was maximal (WD $= 1.4578$) and show the results in Fig. 13. In this case, TACTiS clearly fails to estimate an upwards trend in the median of the ground truth distribution, leading to a median that is not coherent with the bounds of the gap. Fortunately, tasks where TACTiS performs at this level are rare, as supported by the distribution of Wasserstein distances.

In conclusion, the aforementioned results clearly show the ability of TACTiS to perform accurate probabilistic interpolation. It is thus safe to conclude that TACTiS is sufficiently flexible to address both forecasting and interpolation tasks.

D.3. Can TACTiS Learn from Unaligned/Non-Uniformly Sampled Time Series?

The main experiments, presented in §5.2, are based on datasets containing only aligned and uniformly-sampled time series. As such, they cannot determine whether TACTiS can take advantage of its transformer architecture to support unaligned or non-uniformly sampled time series. We use this section to bring evidence that TACTiS is indeed compatible with such time series, using an experiment with a small simulated dataset.

Dataset We consider two independent univariate time series, each generated by adding a random walk to a sinusoidal signal, with the second series having a twice-larger random walk amplitude than the first. From this ground truth, we select a sample of non-uniformly spaced points by selecting one point in each window of 10 time steps in the ground truth, uniformly at random. These points are revealed to the model and the others are hidden. To produce unaligned time series, this procedure is performed independently for each series.

Implementation details To make our implementation of TACTiS compatible with this setting, we had to use the index of each time point in the ground truth time series to generate the positional encodings. The development of positional encodings better suited for temporal data, e.g., that are generated using a real-valued time stamp, goes beyond the scope of this work.

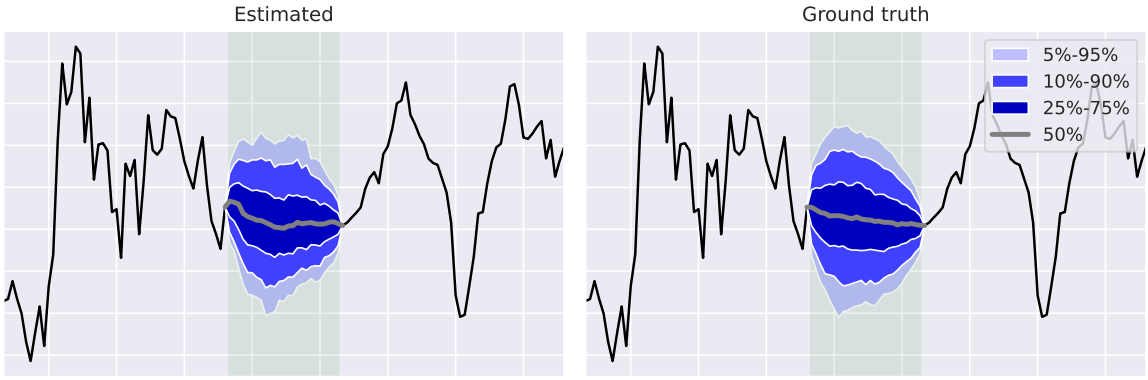


Figure 11. Interpolation - Good performance. Comparison of the interpolation distribution estimated by TACTiS (left) and the ground truth distribution (right) for the task where TACTiS most faithfully approximates the Oracle, as measured by the Wasserstein distance between energy score distributions (WD= 0.0009). The colors correspond to confidence intervals of the distribution. All distributions are estimated using 1000 samples.

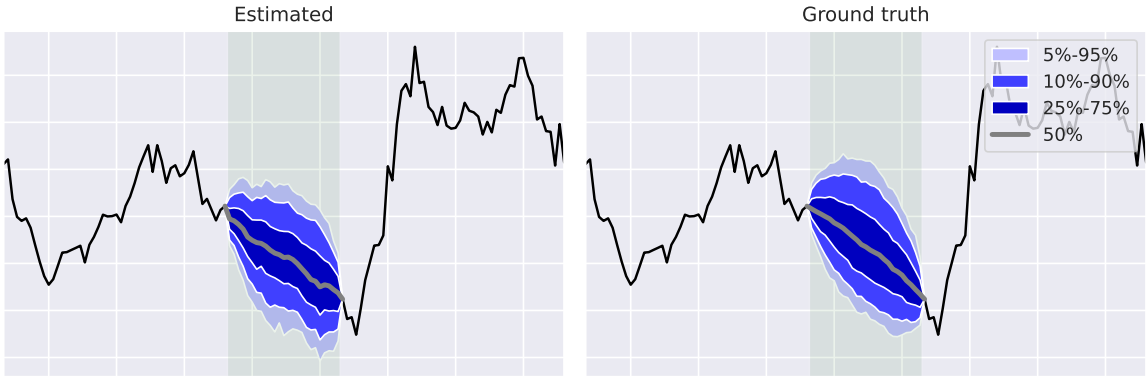


Figure 12. Interpolation - Average performance. Comparison of the interpolation distribution estimated by TACTiS (left) and the ground truth distribution (right) for a task that is representative of the average performance of TACTiS, as measured by the Wasserstein distance between energy score distributions (WD= 0.0388). The colors correspond to confidence intervals of the distribution. All distributions are estimated using 1000 samples.

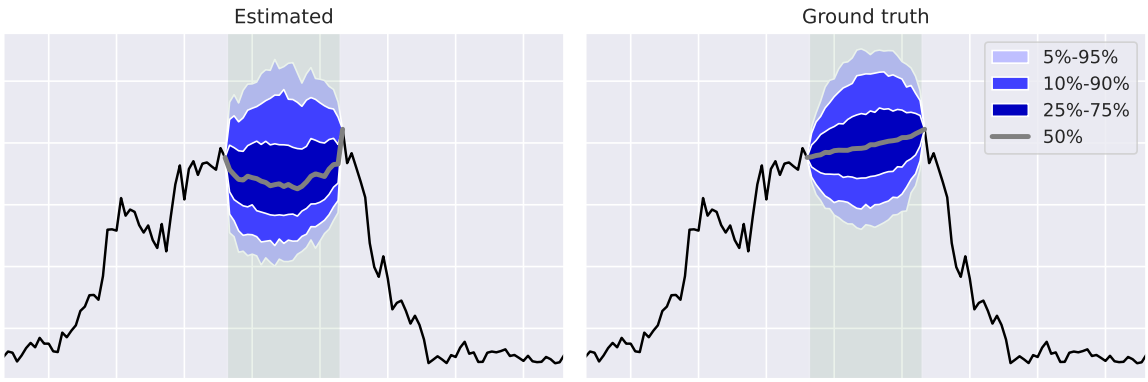


Figure 13. Interpolation - Poor performance. Comparison of the interpolation distribution estimated by TACTiS (left) and the ground truth distribution (right) for the task where TACTiS least faithfully approximates the Oracle, as measured by the Wasserstein distance between energy score distributions (WD= 1.4578). The colors correspond to confidence intervals of the distribution. All distributions are estimated using 1000 samples.

Table 11. Hyperparameters for TACTiS for the interpolation experiment.

	Hyperparameter	Selected value
Model	Encoder transformer embedding size (per head) and feed forward network size	8
	Encoder transformer number of heads	2
	Encoder number of transformer layers	2
	Encoder time series embedding dimensions	5
	Decoder MLP number of layers	1
	Decoder MLP hidden dimensions	8
	Decoder transformer number of heads	2
	Decoder transformer embedding size (per head)	8
	Decoder number transformer layers	2
	Decoder number of bins in conditional distribution	20
	Decoder DSF number of layers	2
	Decoder DSF hidden dimensions	8
	Dropout	0
Data	Normalization	Standardization
	History length to prediction length ratio (before interpolation range)	2
	History length to prediction length ratio (after interpolation range)	2
Training	Optimizer	RMSprop
	Learning rate	10^{-3}
	Batch size	64
	Weight decay	10^{-5}
	Gradient clipping	10^{3}

However, any developments could be directly integrated in TACTiS in a plug-and-play fashion.

Results and conclusions TACTiS was trained to forecast windows of 10 time points, given a history of the same length. The hyperparameters used for the model are given in Tab. 12. In Fig. 14, we show an example forecast produced by TACTiS on this data. While this is by no means a thorough study of the performance of TACTiS in this challenging setting, it does confirm that the model can indeed work with multivariate time series that are unaligned and sampled at various frequencies.

D.4. Ablation Study

In this section, we attempt to measure the relative impact of the following key components of TACTiS-TT on the quality of its predictions:

1. The self-attention layers in the encoder,
2. The use of Deep Sigmoidal Flows to estimate marginal distributions in the decoder, and
3. The use of attentional copulas to model correlations in the decoder.

To achieve this, we compare TACTiS-TT to two ablations:

- TACTiS-IC: replacing the attentional copula with a trivial one, which measures the importance of (3), and
- TACTiS-GC: replacing the entire decoder with the Gaussian-copula-like decoder of GPVar (Salinas et al., 2019), which measures the importance of (2-3).

Comparing these two ablations with TACTiS-TT also allows a measure of the relative importance of (1). For example, if both ablations perform just as well as TACTiS-TT, we may conclude that (1) is the main driver of the model's performance.

Note that it would be challenging to ablate the marginal flows (2) directly without affecting the quality of the learned copula. For example, suppose that we replaced the marginal flows with an empirical CDF (ECDF), as used in Salinas et al. (2019). Training via maximum likelihood would push the attentional copula to encode how the univariate forecasts deviate from the ECDF, which would prevent it from converging to a valid copula. Hence, we abstain from conducting such an experiment.

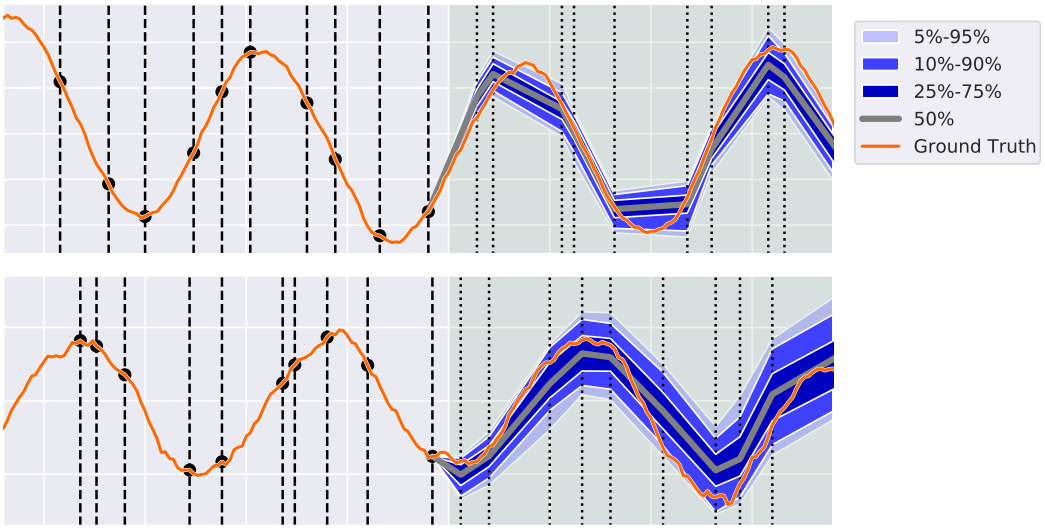


Figure 14. Unaligned and non-uniformly sampled time series. Demonstration of the forecasting ability of TACTiS-TT on two independent time series sampled at unaligned and non-uniform time points. The orange line represents the ground truth of the process to be forecasted. The dashed vertical lines and black dots represent the data points given to the model (history), while the dotted lines represent the time points at which the model is asked to produce forecasts.

Table 12. Hyperparameters for TACTiS for the unaligned/non-uniformly sampled experiment.

	Hyperparameter	Selected value
Model	Encoder transformer embedding size (per head) and feed forward network size	8
	Encoder transformer number of heads	2
	Encoder number of transformer layers	2
	Encoder time series embedding dimensions	5
	Decoder MLP number of layers	1
	Decoder MLP hidden dimensions	8
	Decoder transformer number of heads	2
	Decoder transformer embedding size (per head)	8
	Decoder number transformer layers	2
	Decoder number of bins in conditional distribution	20
	Decoder DSF number of layers	2
	Decoder DSF hidden dimensions	8
	Dropout	0
Data	Normalization	Standardization
	History length to prediction length ratio	1
Training	Optimizer	RMSprop
	Learning rate	10^{-3}
	Batch size	256
	Weight decay	10^{-5}
	Gradient clipping	10^{3}

D.4.1. TACTiS-GC: Using ECDFs and a Gaussian Copula in the Decoder

Here, we evaluate the TACTiS-GC ablation, where we replace the marginal flows with ECDFs and the attentional copula with a low-rank Gaussian copula, as used in Salinas et al. (2019). As is done in Salinas et al. (2019), the ECDFs are computed, per variable, using the history of each sample and are composed with the inverse CDF of normal distribution of zero mean and unit variance. Since this ablation adds new components to the model, which come with their own hyperparameters, we

re-run the hyperparameter search using the ranges listed in Tab. 13. Of note, we include the use of the ECDF as one of many possible data normalization methods in the hyperparameter search (denoted CDF in the table). It can thus be bypassed if this leads to more accurate predictions, in which case the learned joint distribution is simply a low-rank Gaussian.

Table 13. Possible hyperparameters for TACTiS-GC. $^{\mathbf{e}}$, $^{\mathbf{f}}$, $^{\mathbf{k}}$, $^{\mathbf{s}}$, and $^{\mathbf{t}}$ respectively indicate the optimal hyperparameters for `electricity`, `fred-md`, `kdd-cup`, `solar-10min`, and `traffic`.

	Hyperparameter	Possible values
Model	Encoder transformer embedding size (per head) and feed forward network size	$8^{\mathbf{f}}$, $16^{\mathbf{et}}$, $24^{\mathbf{ks}}$
	Encoder transformer number of heads	$1^{\mathbf{t}}$, $2^{\mathbf{efks}}$
	Encoder number of transformer layers pairs	$1^{\mathbf{ef}}$, 2, $4^{\mathbf{kst}}$
	Encoder time series embedding dimensions	$5^{\mathbf{efkst}}$
	Decoder MLP number of layers	$1^{\mathbf{s}}$, $2^{\mathbf{kt}}$, $4^{\mathbf{ef}}$
	Decoder MLP hidden dimensions and rank of the low-rank approximation	$8^{\mathbf{e}}$, $16^{\mathbf{kt}}$, $24^{\mathbf{fs}}$
	Dropout	$0^{\mathbf{est}}$, $0.1^{\mathbf{f}}$, $0.01^{\mathbf{k}}$
Data	Normalization	None$^{\mathbf{st}}$, Standardization$^{\mathbf{e}}$, CDF$^{\mathbf{fk}}$
	History length to prediction length ratio	$1^{\mathbf{f}}$, $2^{\mathbf{et}}$, $3^{\mathbf{ks}}$
Training	Optimizer	RMSprop$^{\mathbf{efkst}}$
	Learning rate	$(10^{-4})^{\mathbf{s}}$, $(10^{-3})^{\mathbf{efkt}}$, (10^{-2})
	Weight decay	0, $(10^{-5})^{\mathbf{ef}}$, $(10^{-4})^{\mathbf{k}}$, $(10^{-3})^{\mathbf{st}}$
	Gradient clipping	$(10^{3})^{\mathbf{efkst}}$, (10^{4})

Tabs. 14 and 15 show the performance of this ablation in comparison with TACTiS-TT. We observe that it performs worse on two datasets: `electricity` and `traffic`, slightly worse on one: `fred-md`, and slightly better on two: `kdd-cup` and `solar-10min`. It should not be surprising that, in some contexts, TACTiS-GC performs well. The low-rank Gaussian copula limits the learned correlations to stem from a small number of latent variables, which may be particularly well suited in some situations. For example, in `solar-10min`, it may help forecast the amount of solar energy produced by nearby stations which operate in similar weather conditions. Nonetheless, since the differences by which TACTiS-GC beats TACTiS-TT are much smaller than those by which it loses, we conclude that TACTiS-TT is generally superior.

Table 14. Means and standard errors of the CPRS-Sum metrics for the ablation experiments.

Model	`electricity`	`fred-md`	`kdd-cup`	`solar-10min`	`traffic`
TACTiS-TT	0.021 ± 0.005	0.042 ± 0.009	0.237 ± 0.013	0.311 ± 0.061	0.071 ± 0.008
TACTiS-GC	0.054 ± 0.011	0.050 ± 0.011	0.214 ± 0.008	0.283 ± 0.023	0.125 ± 0.010
TACTiS-IC	0.022 ± 0.005	0.042 ± 0.009	0.303 ± 0.009	0.381 ± 0.068	0.080 ± 0.011

Table 15. Means and standard errors of the energy score metrics for the ablation experiments.

Model	`electricity` $\times 10^4$	`fred-md` $\times 10^5$	`kdd-cup` $\times 10^3$	`solar-10min` $\times 10^2$	`traffic` $\times 10^0$
TACTiS-TT	5.42 ± 0.57	8.18 ± 1.83	2.93 ± 0.22	2.88 ± 0.23	3.10 ± 0.13
TACTiS-GC	8.70 ± 0.89	9.55 ± 1.95	2.65 ± 0.17	2.94 ± 0.05	4.99 ± 0.02
TACTiS-IC	5.32 ± 0.54	8.40 ± 1.69	5.24 ± 0.42	9.14 ± 1.25	10.69 ± 1.04

D.4.2. TACTiS-IC: Using a Trivial Copula Instead of an Attentional Copula

Here, we evaluate the TACTiS-IC ablation in which we replace the attentional copula with a trivial copula, where all variables are independent and their distribution is $U_{[0,1]}$. The goals of this ablation are twofold: 1) measure the contribution of the marginal flows vs. the dependencies learned by the attentional copulas in the quality of predictions, and 2) make sure that the learned attentional copulas are non-trivial and that they contribute to the forecasts. Since this ablation does not add new hyperparameters or model parameters, we reuse the trained TACTiS-TT models to conduct this experiment.

Tabs. 14 and 15 show the performance of this ablation in comparison to TACTiS-TT. Interestingly, the CRPS-Sum and energy score metrics paint two different pictures: for CRPS-Sum, the results are quite similar with only two datasets showing a clear advantage for TACTiS-TT: `kdd-cup` and `solar-10min`. However, for the energy score, there are three datasets for which TACTiS-TT dominates TACTiS-IC: `kdd-cup`, `solar-10min`, and `traffic`. This is consistent with CRPS-Sum having a lower discrimination ability than the energy score, as discussed in Koochali et al. (2022).

The lack of performance degradation on the CRPS-Sum metric shows that learning very good marginal distributions without good correlations is sufficient to achieve good CRPS-Sum values. In contrast, the energy score is much better at revealing the lack of learned correlations. Nonetheless, there are two datasets: `electricity` and `fred-md`, where we do not see a contrast in energy score, but for which we know that the attentional copula successfully learned non-trivial correlations (see §E.3). This suggests that either the energy score is also dominated, to some extent, by fitting the marginal distributions very well or that the attentional copula's contribution to forecast quality is less than that of properly fitting the marginals for these datasets.

D.4.3. Relative Importance of TACTiS-TT Key Components

When comparing the ablations results in Tabs. 14 and 15 with our benchmarks in Tabs. 1 and 9, we see that both TACTiS-TT and its ablations, TACTiS-GC and TACTiS-IC, are quite competitive with the state of the art. This suggests that the shared portion of their architecture, i.e., the self-attention-based encoder (1), is the primary driver of their good performance. Now, what can we say about the importance of the marginal flows (2) and the attentional copula (3)?

Our results indicate that these components are important drivers of the good performance but that their individual importance depends on the dataset under consideration. The importance of the marginal flows (2) is revealed by the `electricity` and `fred-md` datasets, where TACTiS-IC outperforms TACTiS-GC. The importance of modelling dependencies between random variables, which is the role of the attentional copula (3), instead of only learning their marginals, is revealed by the `kdd-cup` and `solar-10min` datasets, where the TACTiS-IC fails in comparison to TACTiS-TT and TACTiS-GC. Finally, the joint importance of the marginal flows (2) and the attentional copula (3), is revealed by the `traffic` dataset, where TACTiS-TT outperforms TACTiS-IC and TACTiS-GC.

In summary, we conclude that the main driver of the performance of TACTiS-TT's good performance is its transformer-based architecture, but that each of the components of its decoder, namely the marginal flows and the attentional copula, play a key role in achieving state-of-the-art results on some datasets.

E. A Deeper Dive into TACTiS Models

E.1. Some Good and Bad Forecasts

In this section, we present a few examples of the probabilistic forecasts generated by TACTiS-TT in our forecasting experiments. Due to the dimensionality of the datasets, it would be infeasible to present all forecasts, so we selected a few examples to demonstrate both the strength and weakness of TACTiS-TT. For each dataset, we thus hand-picked four forecasts that were *particularly good* from the backtesting experiment with the lowest CRPS-Sum and four *particularly bad* forecasts from the backtesting experiment with the highest CRPS-Sum.

The `electricity` dataset Fig. 15 shows some particularly good forecasts for the `electricity` dataset. The top- and bottom-left forecasts show that the model could predict that the time series would likely see a sudden increase after a zero-valued gap. The top- and bottom-right forecasts show the model's ability to follow the daily periodicity of the data. The top-right forecast is particularly impressive since the model was able to foresee the higher demand that would occur on Monday (2014-12-15), following lower values for Saturday (2014-12-13) and Sunday (2014-12-14), and this, even though our positional encodings do not include day-of-week information.

Fig. 16 shows some particularly bad forecasts for the `electricity` dataset. In the top-left forecast, the distribution of forecasted values is near-uniform, even though the data in this series only take on three possible values. This illustrates a limitation of the marginal flows, which may poorly fit discrete distributions, especially when only a small number of series in the dataset contain discrete patterns. In the top-right forecast, the model is extremely confident that the series would obey the same pattern as in the previous days, missing the change in dynamics. The bottom-left one is similar; the model believes that the forecasted day would be similar to the previous ones, failing to capture the increase that occurs from day to day. The bottom-right forecast also shows an example where the model assumes that the series has daily periodicity, even though the history reveals this is not the case. Note that using positional encodings especially designed for temporal data may help circumvent these issues.

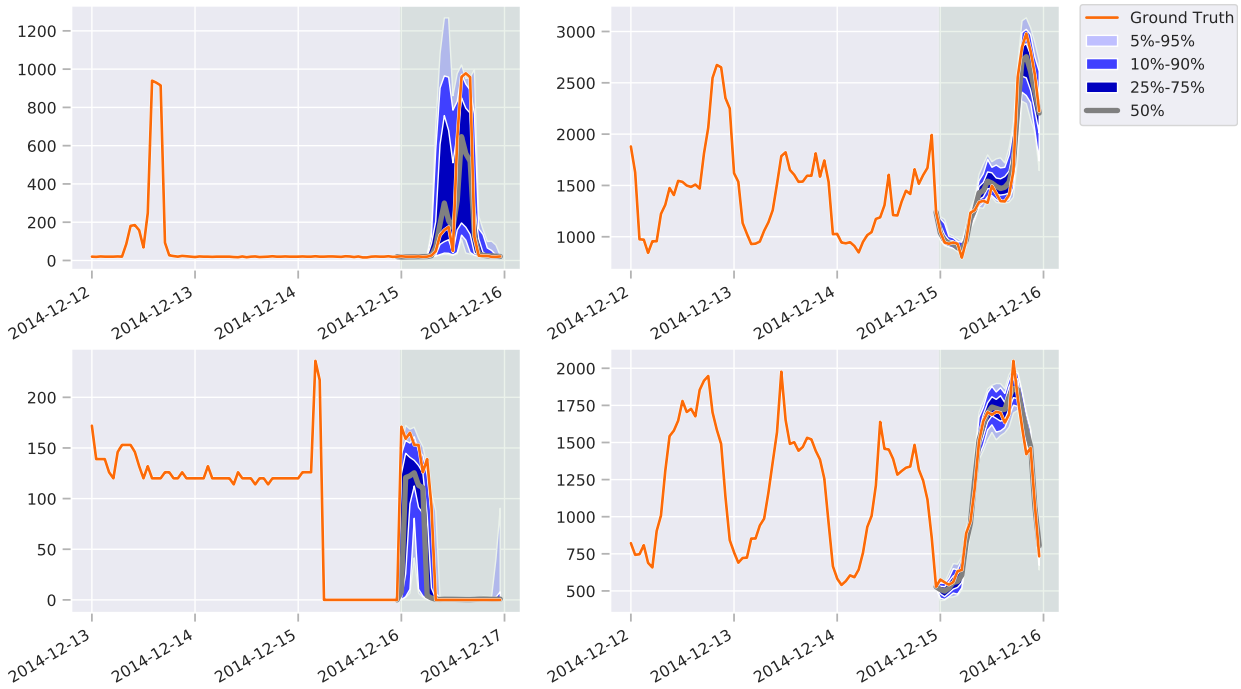


Figure 15. Hand-picked example forecasts for the `electricity` dataset. These examples have been selected to show particularly **good** forecasts. The historical ground truth shown is the one that was made available to the model. The data is from 2014, so December 15th is a Monday.

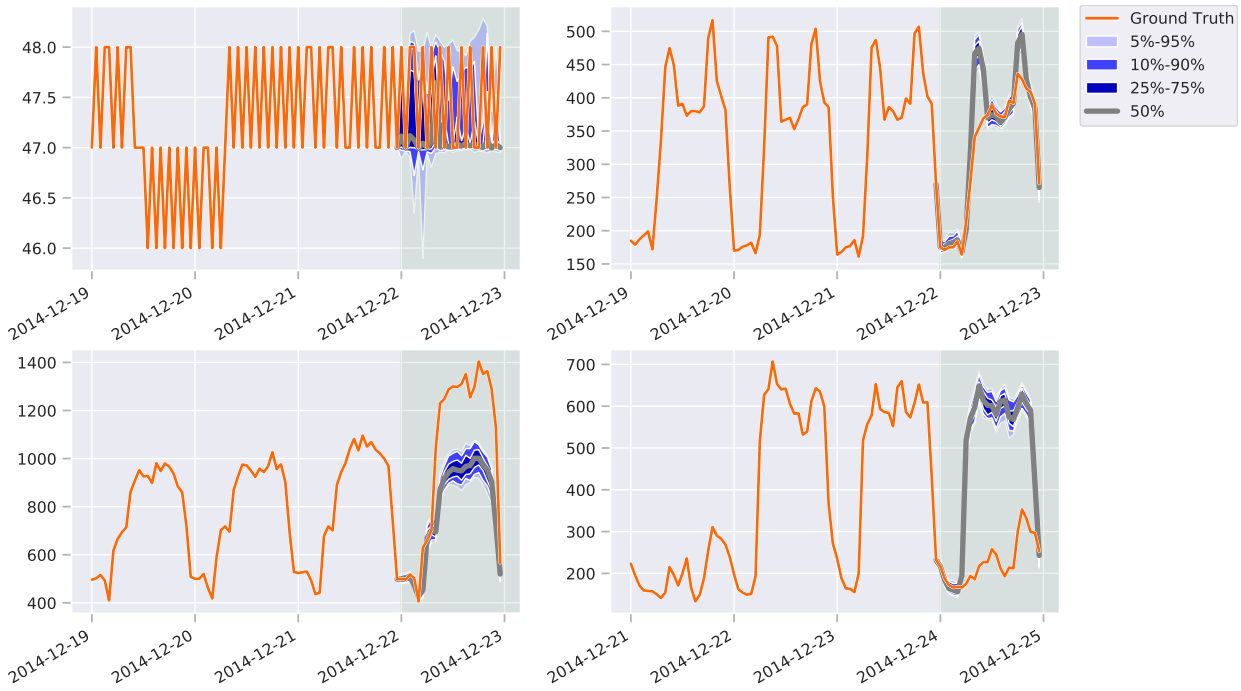


Figure 16. Hand-picked example forecasts for the `electricity` dataset. These examples have been selected to show particularly **bad** forecasts. The historical ground truth shown is the one that was made available to the model.

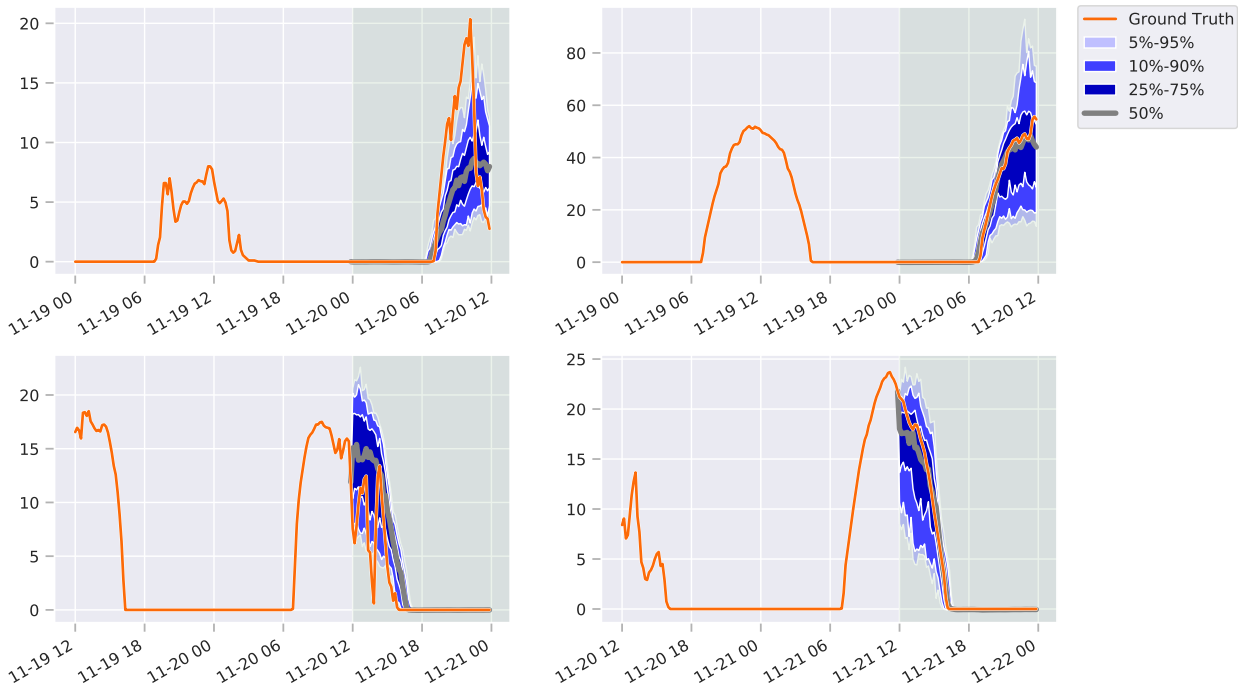


Figure 17. Hand-picked example forecasts for the `solar-10min` dataset. These examples have been selected to show particularly **good** forecasts. The historical ground truth shown is the one that was made available to the model.

The `solar-10min` dataset Fig. 17 shows some particularly good forecasts for the `solar-10min` dataset. This dataset has the particularity of having sudden variations (increases or decreases) between consecutive time points. The forecasts we present in this figure all seem to account for this characteristic by attributing significant likelihood to a wide range of values in their forecasts. The bottom-left and -right forecasts are particularly interesting since their distributions are oddly shaped: uncertainty is maximal at the beginning of the forecast and decreases afterwards. Again, this seems to account for the stochasticity that exists within the process while also accounting for the fact that periods of non-zero value always revert to zero periodically.

Fig. 18 shows some particularly bad forecasts for the `solar-10min` dataset. All of the forecasts have the same issue: the model often predicts a peak in values around mid-day and, when this happens, fails to attribute high likelihood to values as low as those observed in the history.

The `fred-md` dataset Fig. 19 shows some particularly good forecasts for the `fred-md` dataset. These show that the model can be applied to data with various trends. Notice how the forecasts' variance gradually increases as we move away from the history, indicating that the model properly accounts for the increasing uncertainty arising from the compounding effects of more and more random events. Furthermore, the ground-truth values stay within the forecasted range while sometimes deviating from the median, suggesting that the variance is not overestimated.

Fig. 20 shows some particularly bad forecasts for the `fred-md` dataset. The top-left and -right forecasts show that the model relied too much on the historical trend and either overreacted to a blip in the data (from 2012-10 to 2012-12 in the top-left) or missed a change in the trend (top-right). Further, in the bottom-left and -right forecasts, the model completely fails to account for radical changes in the behaviour of the series.

The `kdd-cup` dataset Fig. 21 shows some particularly good forecasts for the `kdd-cup` dataset. All four are quite similar in that the model predicts a relatively uniform distribution on a range of possible values, which seems to be accurate based on the oscillations in the ground truth. Furthermore, we see a gradual increase of variance at time points subsequent to the history, indicating that the model learned that, in this dataset, changes occur gradually rather than suddenly (e.g., in the `solar-10min` dataset).

Fig. 22 shows some particularly bad forecasts for the `kdd-cup` dataset. The top-left forecast shows that the model underestimated the maximum value of the series, not having seen such values in the history. The bottom-left shows the opposite: the model underestimated the odds of the series remaining at relatively constant and low values. The top-right

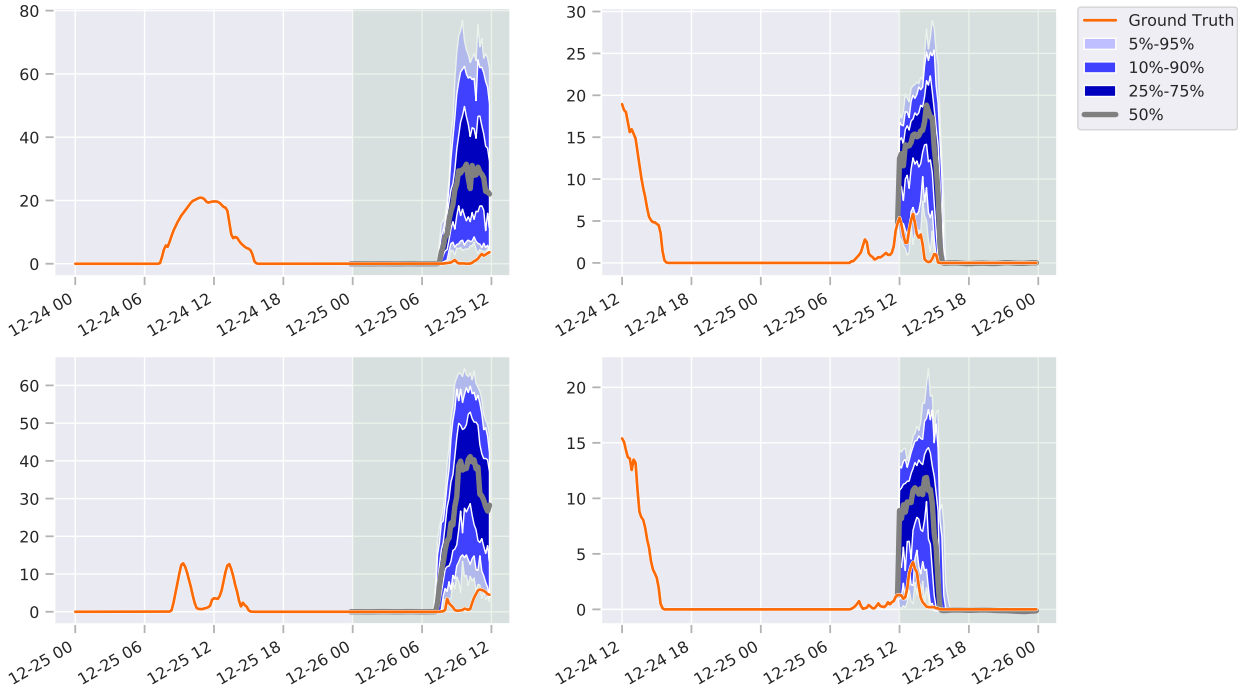


Figure 18. Hand-picked example forecasts for the `solar-10min` dataset. These examples have been selected to show particularly **bad** forecasts. The historical ground truth shown is the one that was made available to the model.

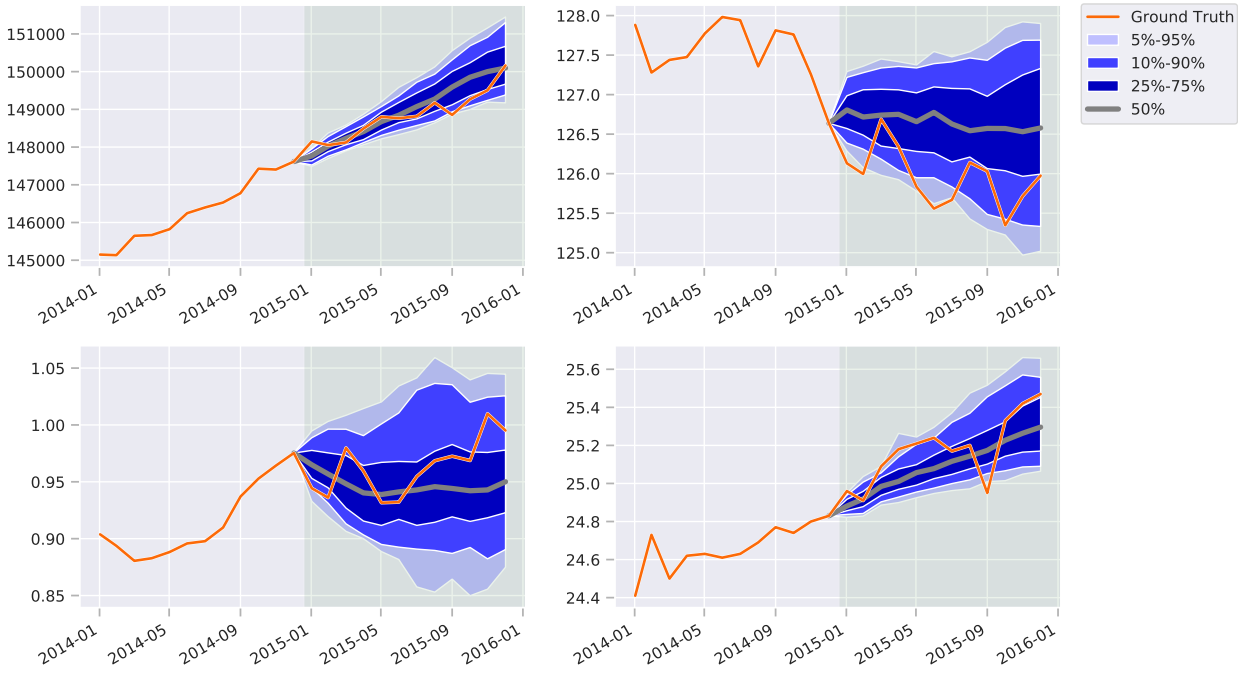


Figure 19. Hand-picked example forecasts for the `fred-md` dataset. These examples have been selected to show particularly **good** forecasts. The historical ground truth shown is the one that was made available to the model.

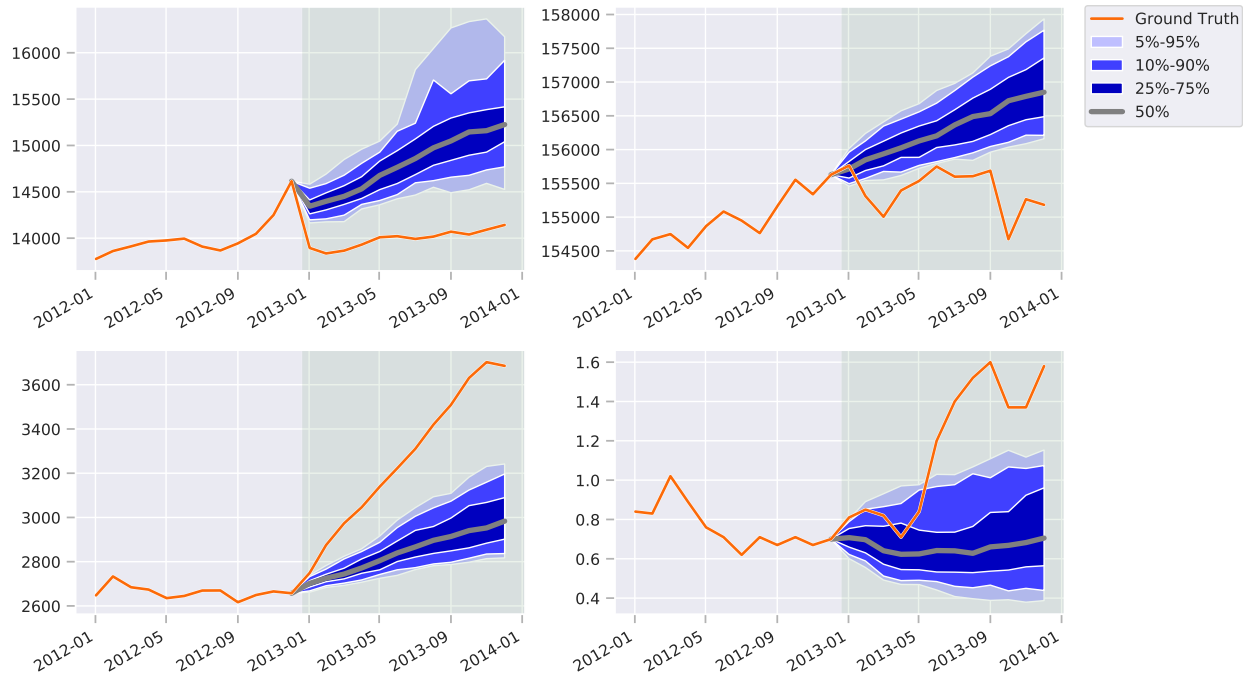


Figure 20. Hand-picked example forecasts for the `fred-md` dataset. These examples have been selected to show particularly **bad** forecasts. The historical ground truth shown is the one that was made available to the model.

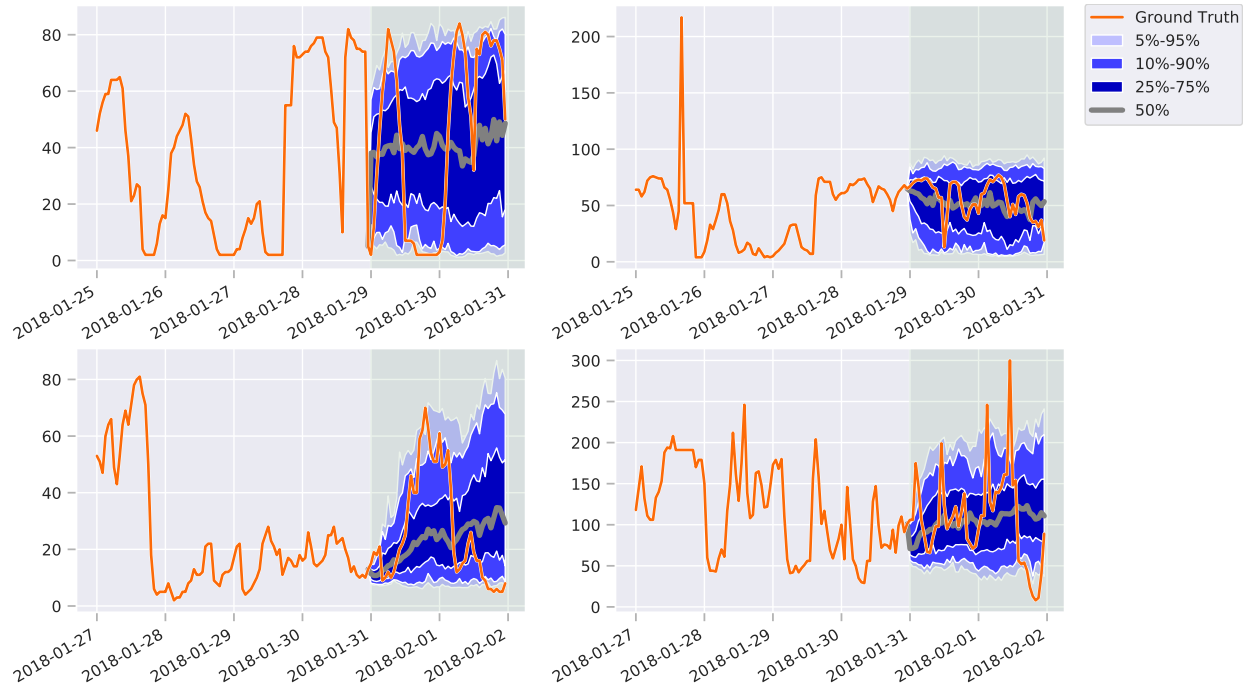


Figure 21. Hand-picked example forecasts for the `kdd-cup` dataset. These examples have been selected to show particularly **good** forecasts. The historical ground truth shown is the one that was made available to the model.

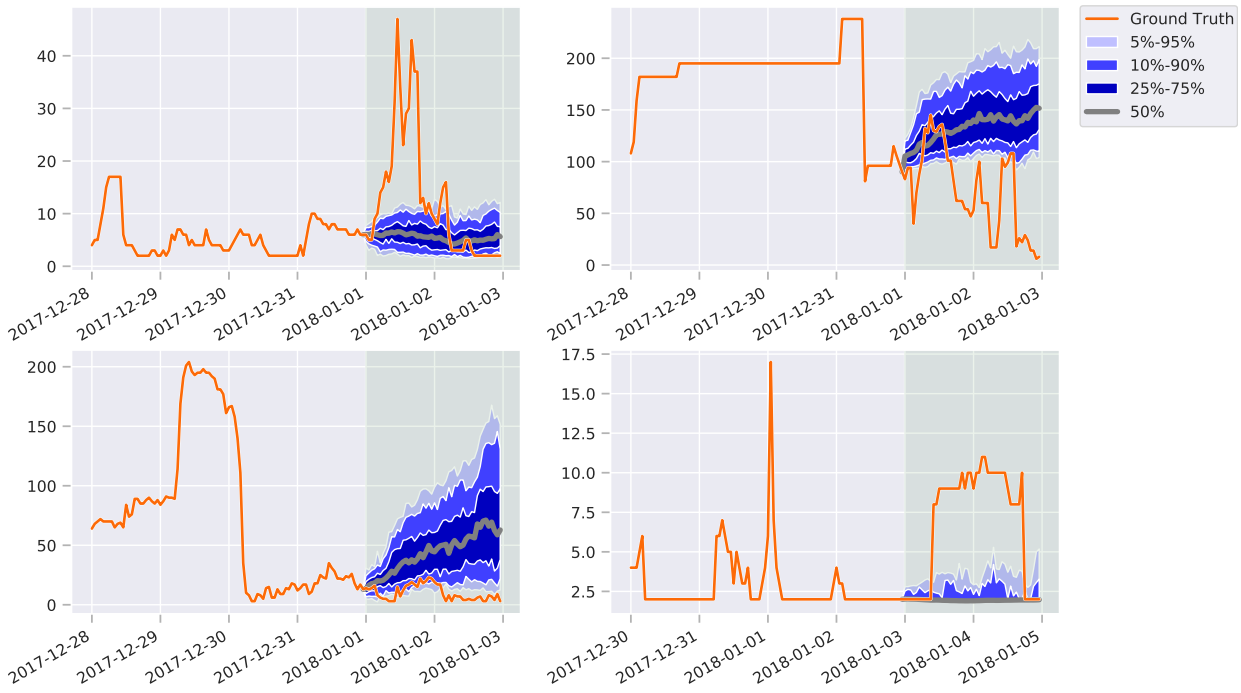


Figure 22. Hand-picked example forecasts for the `kdd-cup` dataset. These examples have been selected to show particularly **bad** forecasts. The historical ground truth shown is the one that was made available to the model.

forecast assumed a slow return to the recent normal, disregarding the possibility of a further decrease. The bottom-right forecast failed to predict that the series could increase as much as it did, even though the recent history showed such increases.

The `traffic` dataset Fig. 23 shows some particularly good forecasts for the `traffic` dataset. The forecast on the top left shows that TACTiS-TT predicted the morning and evening traffic spikes, even though only the morning spike of the previous day was visible in the history. The bottom-left forecast shows the model's uncertainty about whether traffic would return to normal after a seemingly anomalous history with most of the data at zero. The two forecasts on the right are examples where TACTiS-TT was able to recognize a pattern in the data and forecast it with very high confidence. Note that this pattern abounds in the `traffic` dataset for many variables and time stamps, which explains why the model could learn to estimate it so accurately.

Fig. 24 shows some particularly bad forecasts for the `traffic` dataset. In the top-left forecast, the model assumed that the forecasted day would match the previous one since similar historical patterns are typically followed by a repetition of the same pattern, thus missing a significant disruption. For the top-right forecast, the model assumed a quick increase in traffic, ignoring the possibility of a delay before the series deviated from zero. In the bottom-left forecast, the model dismissed again the possibility of an anomaly, which would cause traffic to go to zero. Finally, in the bottom-right forecast, the model did not foresee that the series could reach values considerably greater than those in the history.

E.2. Looking into Learned Marginal Distributions

TACTiS forecasts result from a combination of an attentional copula and the marginal distributions learned for each variable. Since the marginal distributions are significant contributors to the quality of forecasts (as shown in §D.4.2), we use this section to look into some of the learned marginal distributions.

Figs. 25 and 26 show a few hand-picked example marginals generated from the learned models. The marginal cumulative distribution function (CDF) of a variable is directly obtained from $F_{\phi_k}(x_k^{(m)})$, while its probability density function (PDF) is obtained by differentiating the CDF w.r.t. $x_k^{(m)}$. Note that these marginals are normalized using the standardization procedure (see §B.4), so a distribution with a mean of 0 and a variance of 1 would be expected if the model simply took the historical values as the forecast.

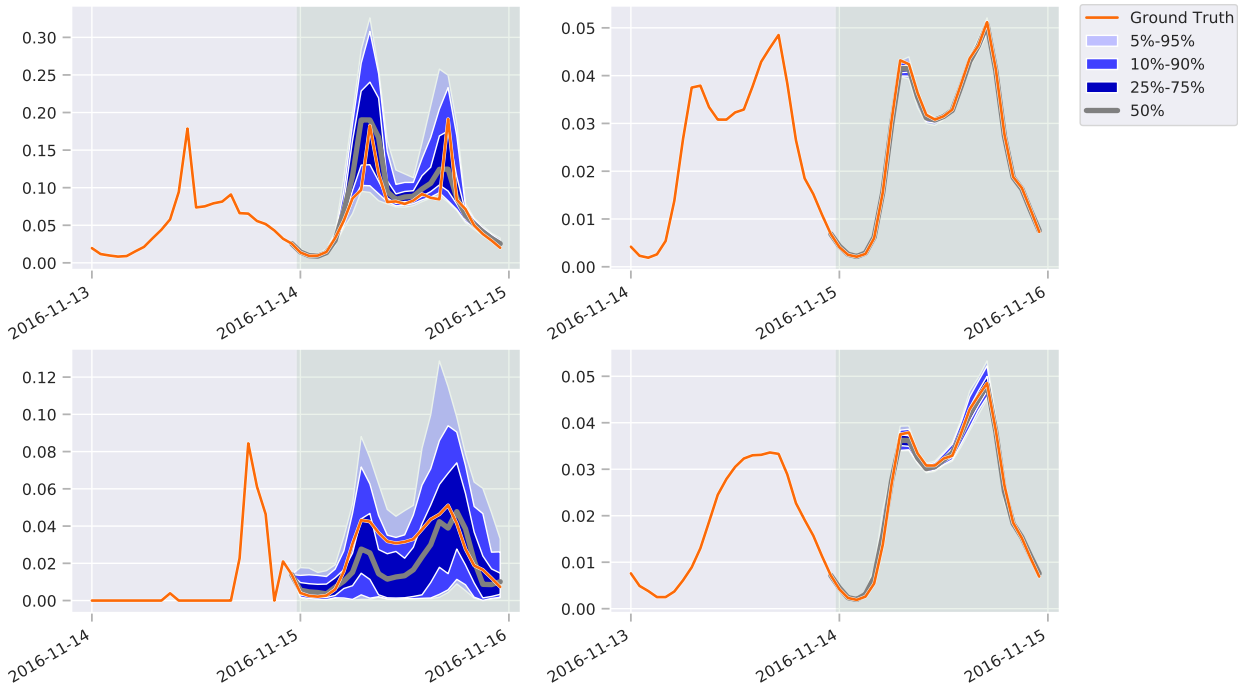


Figure 23. Hand-picked example forecasts for the `traffic` dataset. These examples have been selected to show particularly **good** forecasts. The historical ground truth shown is the one that was made available to the model.

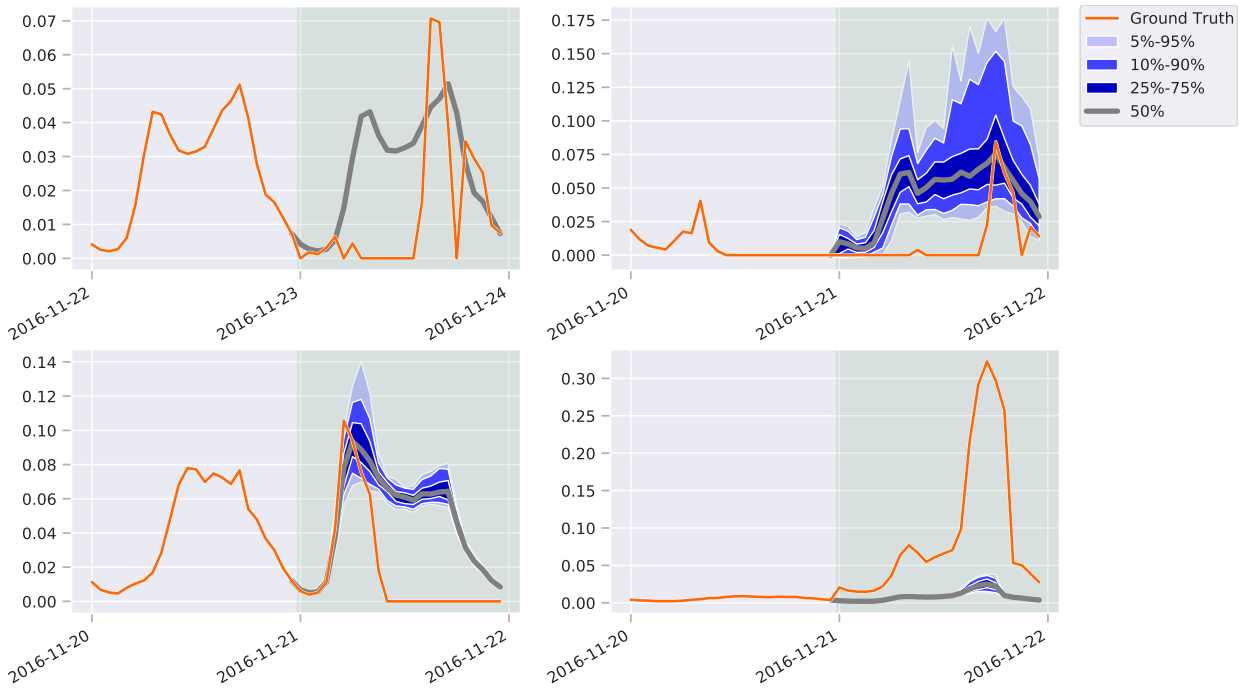


Figure 24. Hand-picked example forecasts for the `traffic` dataset. These examples have been selected to show particularly **bad** forecasts. The historical ground truth shown is the one that was made available to the model.

Unimodal marginals As can be seen in Fig. 25, the marginals for `electricity`, `fred-md`, and `traffic` are all unimodal. The main difference is that the `fred-md` marginal is quite broad, showing that the variable has an extensive range of possible values that are outside the range of historical values. In contrast, the `traffic` marginal is very narrow, demonstrating that the model is quite confident in the value of the variable.

Multimodal marginals As can be seen in Fig. 26, the marginals for `kdd-cup` and `solar-10min` are both multimodal, something that would not have been possible using many common parametric distributions, such as the normal or gamma distributions. Furthermore, both marginals have a significant spike at their minimal value, showing that the model has learned that some variables have a non-zero probability of being exactly at their lower bound in these datasets. For example, in the `solar-10min` dataset, this may be caused by zero solar energy production on a cloudy day.

E.3. Learning Dependencies Between Variables

In §D.4.2, we have shown that the TACTiS-IC ablation, which removes all dependencies between the variables, still leads to decent results for the metrics. This could indicate that TACTiS-TT makes predictions using only its marginal distributions without having learned anything noteworthy in its attentional copula. To verify if this is the case, we can inspect the correlations between variables in the predictive distribution of the model (see § 2.3). If the copula were unused, we would observe zero Pearson product-moment correlation coefficients between all variables. Otherwise, we would observe non-zero values. Rest assured, the following experiments show that TACTiS-TT does learn correlations between variables, making use of its attentional copula.

Intra-series correlations In this first experiment, we inspect the correlations learned between various time steps within the same series (intra-series). To achieve this, we compute the correlation coefficient between the last forecasted time step [13] and all other earlier time steps of the forecast. These are then averaged over the multiple series and the multiple forecasts for each trained model (one per backtesting period and trial). Fig. 27 shows the distribution of these averaged correlation coefficients for each studied dataset. From these, it is clear that TACTiS-TT used its attentional copula to learn intra-series dependencies. As expected, the correlation between time steps decreases with time. An exception for time differences nearing 24 hours can be observed, e.g., in `traffic`. This could be explained by stochastic events impacting multiple days but only at certain times during the day.

Inter-series correlations We now look into the ability of TACTiS-TT to model dependencies between the series (inter-series) using its attentional copula. To achieve this, we use samples taken at the last time step in the forecast to measure inter-series correlations. Figs. 28 to 32 show the average correlation coefficients between all series, averaged over all forecasts, backtesting periods, and trials. While the inter-series correlations tend to be of lesser magnitude than intra-series correlations, TACTiS-TT did indeed learn some of the structure of the datasets. In particular, it isolated two clusters of interdependent variables in `kdd-cup`, which makes sense given that the series were collected in two cities (Beijing and London). Similarly, it was able to recognize that `fred-md` data contains multiple groups of series with strong dependencies inside each group. The `traffic` result is somewhat lacklustre since we expected more spatial correlations for this dataset, which captures road occupancy levels at multiple intersections, some of which must be nearby. This indicates that while TACTiS-TT results are impressive, there is still improvement to be made to learn the dependencies in some of these datasets.

E.3.1. Impact of Bagging on Inter-Series Correlations

In Tab. 2, we have shown that reducing the number of series available at once during training has no negative impact on the quality of the various metrics. However, does it affect the models' ability to learn inter-series dependencies? In this section, we repeat the inter-series correlations experiment for the forecasts obtained by models trained using various bagging sizes. The results for bagging sizes 1, 2, 5, and 10 are shown in Fig. 33. As expected, training with a bagging size of 1 prevents the model from correctly learning inter-series dependencies. This occurs because the model never sees data from multiple series during training; thus, the attention mechanism cannot learn to block the flow of information between two independent series. The result is a matrix where all series are quite correlated; in fact, the correlation values are similar to those in Fig. 27, suggesting that the model may consider all variables to be from the same series. For a bagging size of 2, the correlations between some series start to fade, showing that the attention mechanism is becoming more selective about dependency patterns. Finally, after the bagging size is increased to 5 or 10, the inter-series correlations seem to converge, suggesting that further increasing the bag size would not change the learned correlations.

One concern raised by this experiment is that the energy score could not reveal the highly incorrect correlations learned for a

[13] Selected due to typically being the most variable.

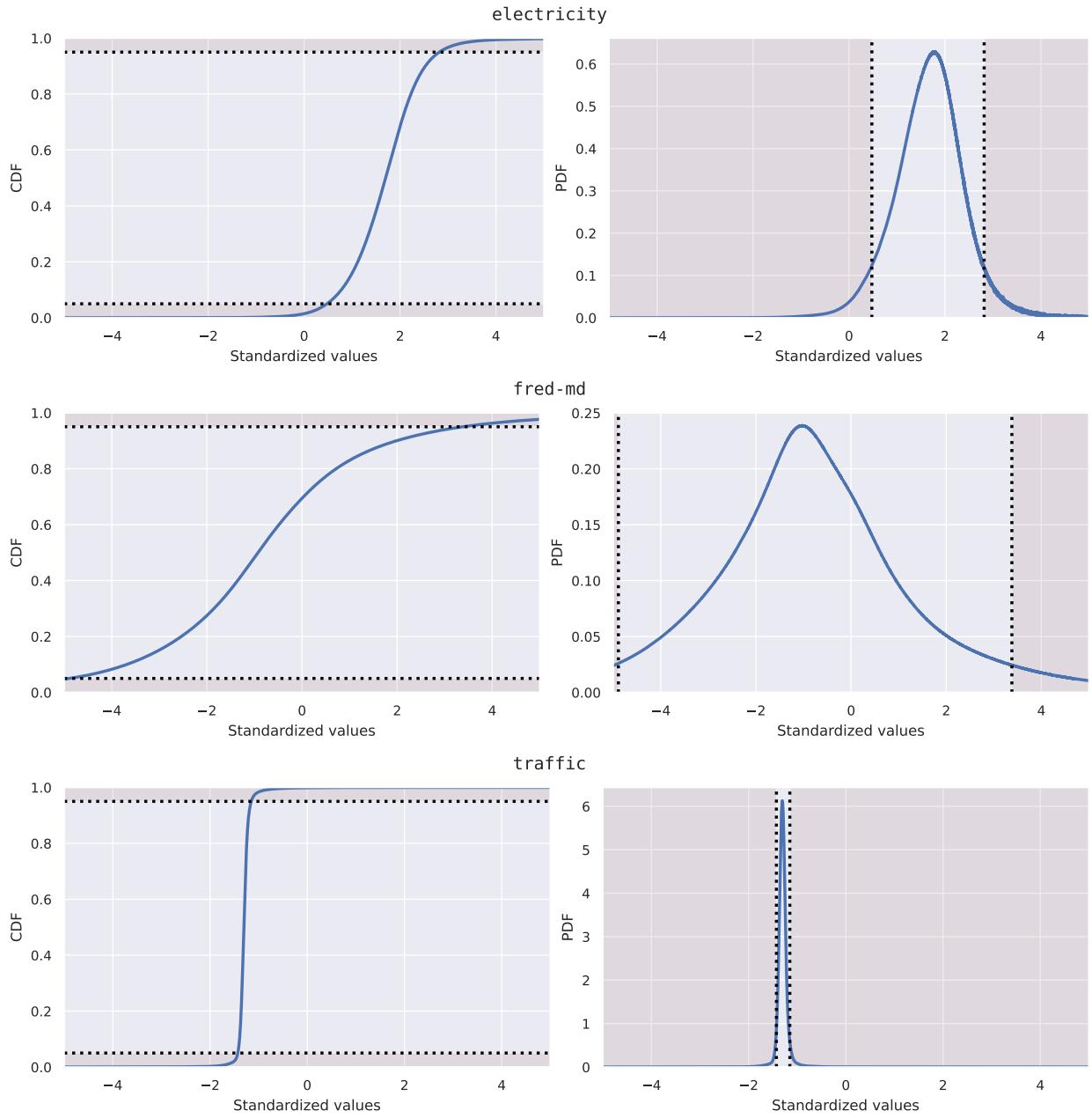


Figure 25. Cumulative distribution functions (CDF) and probability density functions (PDF) of hand-picked example marginals generated by TACTiS-TT for `electricity`, `fred-md`, and `traffic`. These examples have been selected to show various behaviours of unimodal marginals. The red areas represent the part of the distribution which has been removed from the sampling, as mentioned in §B.2.

bagging size of 1. The energy score for a bagging size of 1 is equal to 8.43 ± 1.76, which is very close to the one obtained for a bagging size of 20 (8.30 ± 1.68) and even better than the one obtained for a bagging size of 5 (9.36 ± 1.88). One possible explanation for this lack of sensitivity is that the energy score is less sensitive at high dimensions, as explored in Pinson & Tastu (2013). Further investigation is required to determine whether this is indeed the case in this experiment or if this is due to another cause.

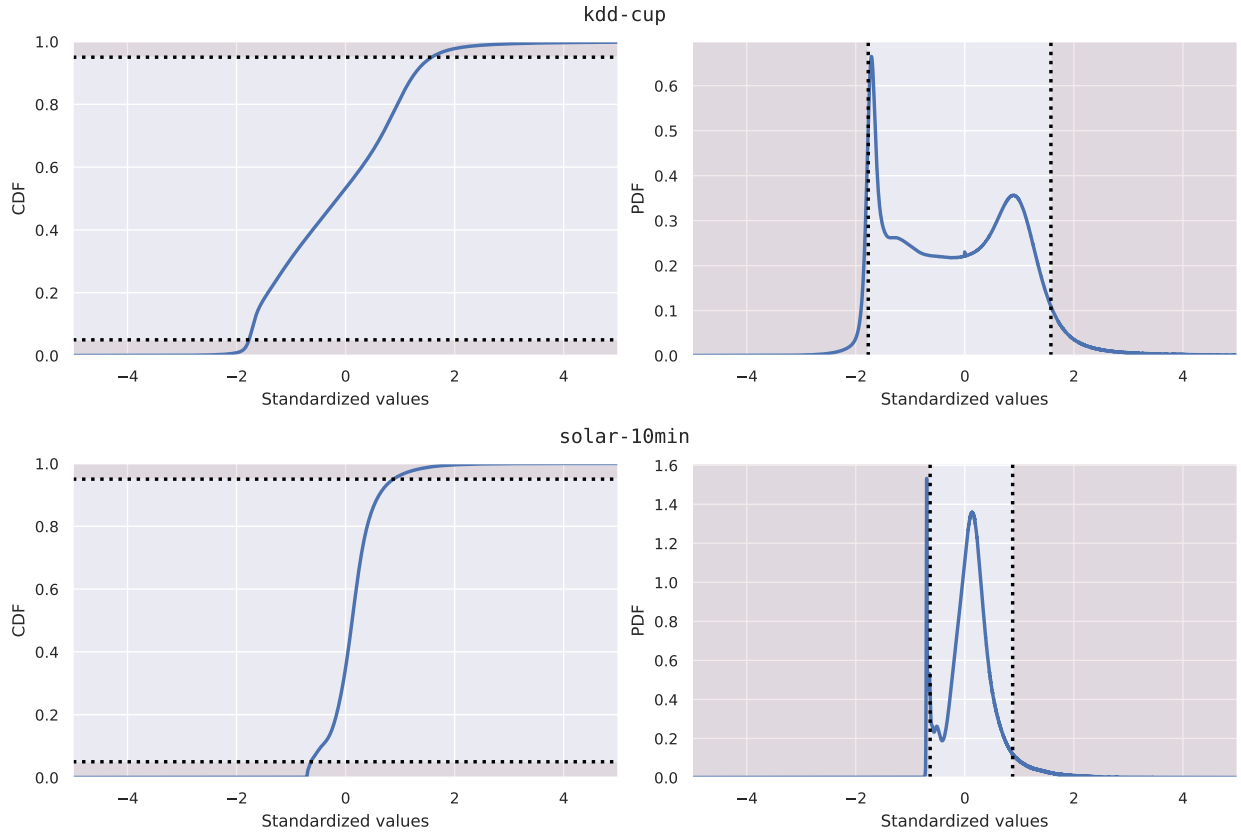


Figure 26. Cumulative distribution functions (CDF) and probability density functions (PDF) of hand-picked example marginals generated by TACTiS-TT for `kdd-cup` and `solar-10min`. These examples have been selected to show various behaviours of multimodal marginals. The red areas represent the part of the distribution which has been removed from the sampling, as mentioned in §B.2.

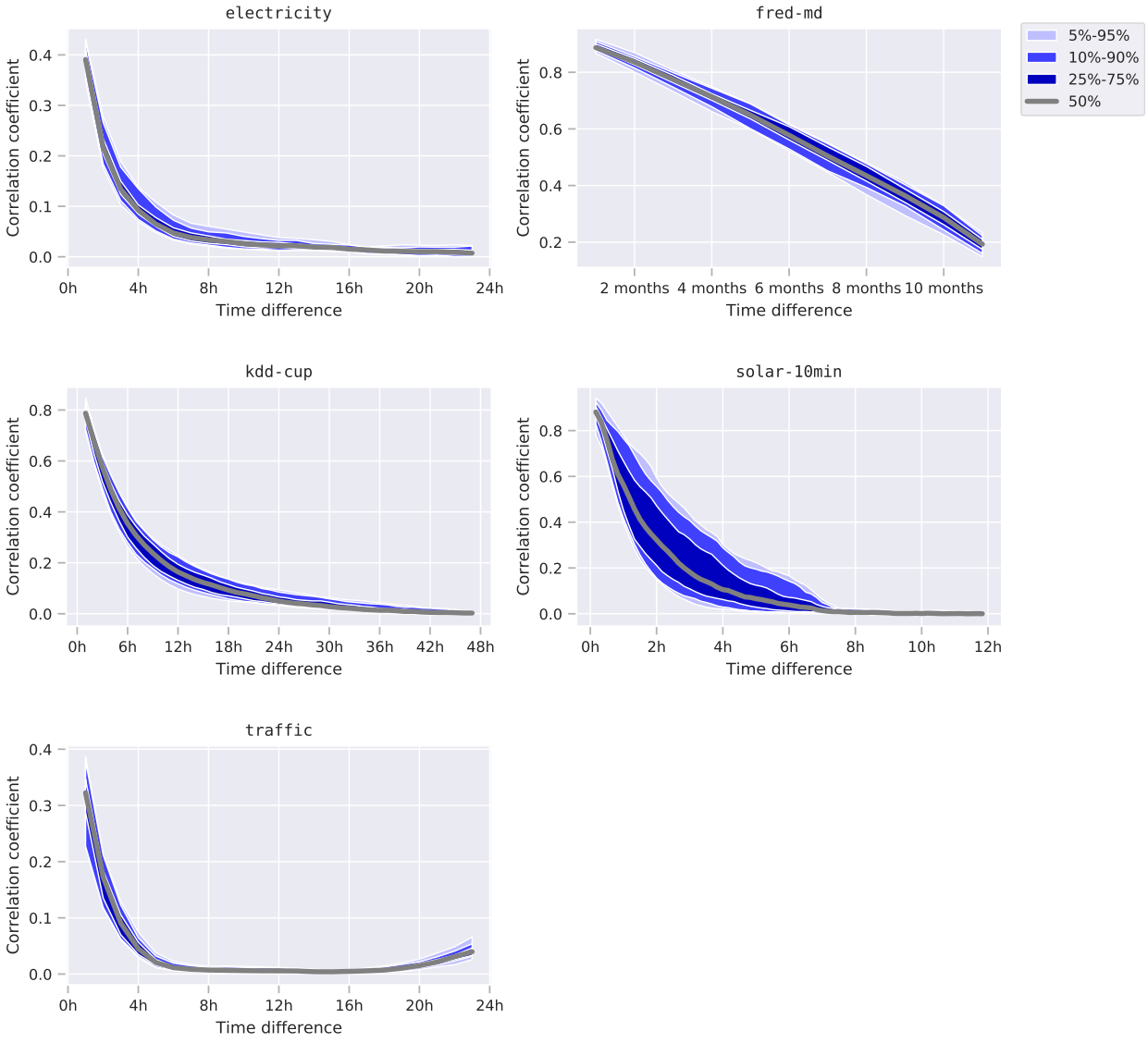


Figure 27. Distribution of the correlation coefficients between the last forecasted time step and prior time steps as a function of the time difference between both. The correlations are averaged over the multiple series and forecasts, and the distribution is taken over the multiple trained models for each dataset.

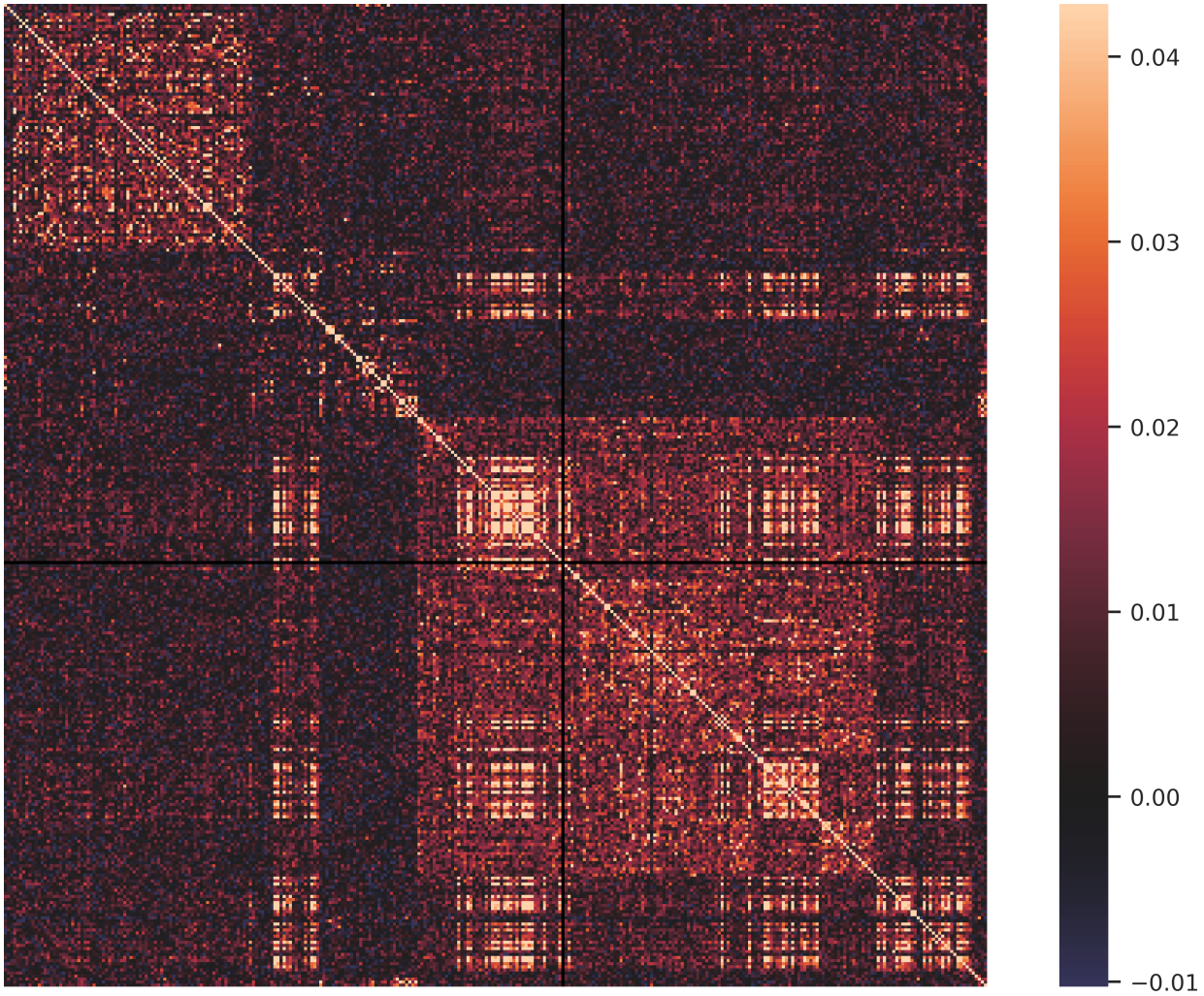


Figure 28. Average correlation coefficients between all series at the last forecasted time step for the `electricity` dataset. The bounds of the colour map are set to the 0.02 and 0.98 quantiles of the data. The black lines are caused by a variable with a constant forecast, thus preventing it from having correlation coefficients.

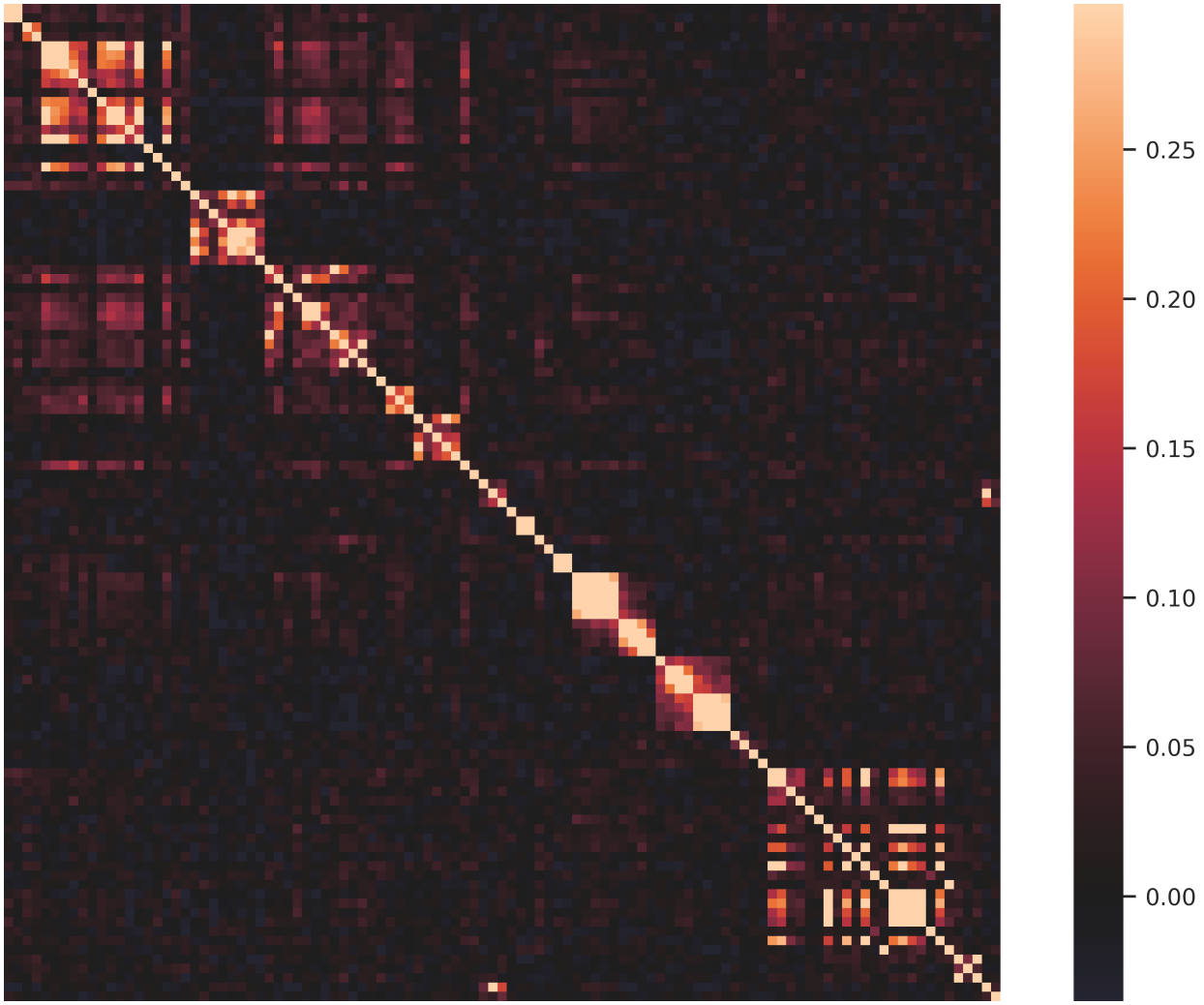


Figure 29. Average correlation coefficients between all series at the last forecasted time step for the `fred-md` dataset. The bounds of the color map are set to the 0.02 and 0.98 quantiles of the data.

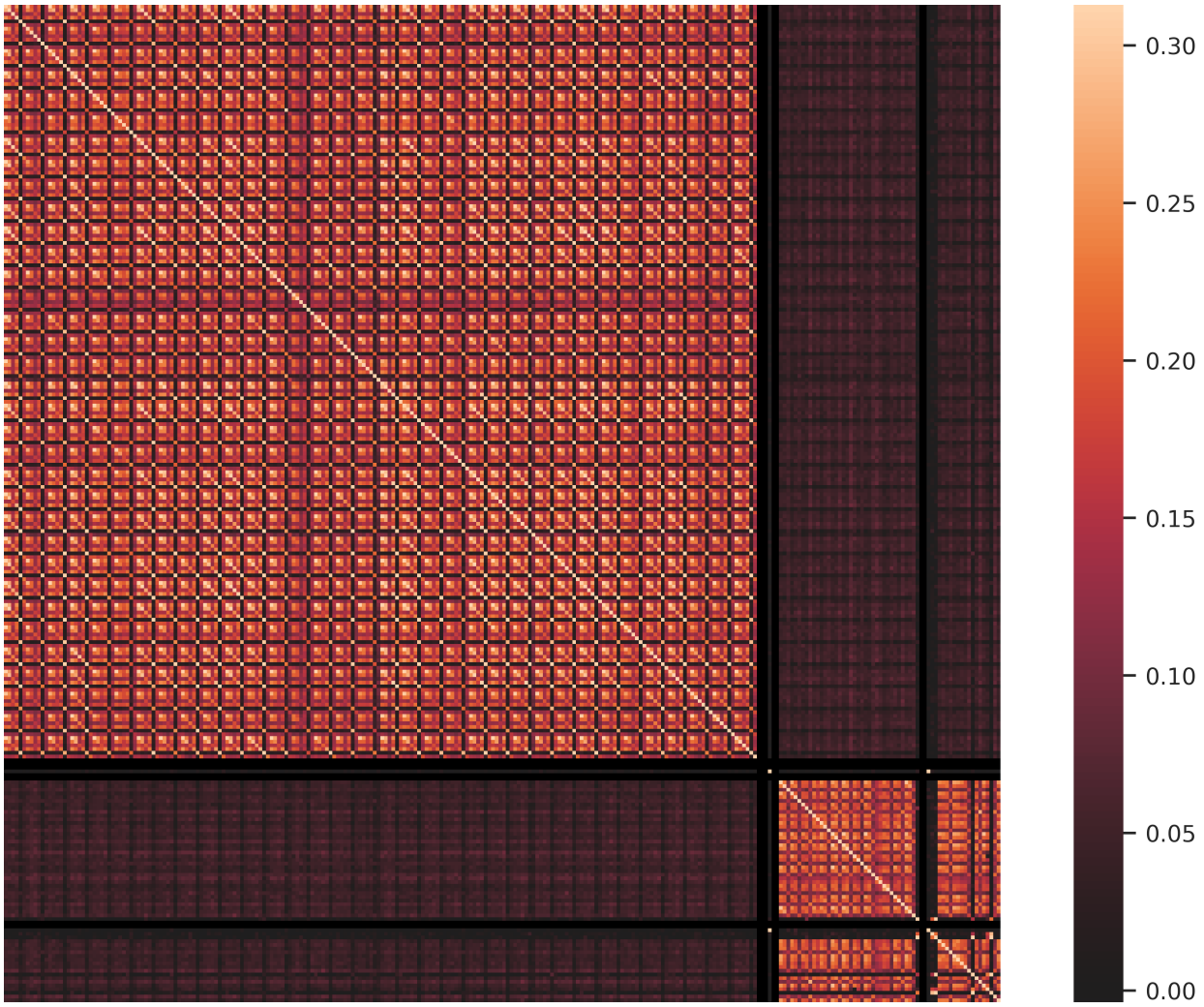


Figure 30. Average correlation coefficients between all series at the last forecasted time step for the `kdd-cup` dataset. The bounds of the colour map are set to the 0.02 and 0.98 quantiles of the data. The black lines are caused by variables with constant forecasts, thus preventing them from having correlation coefficients.

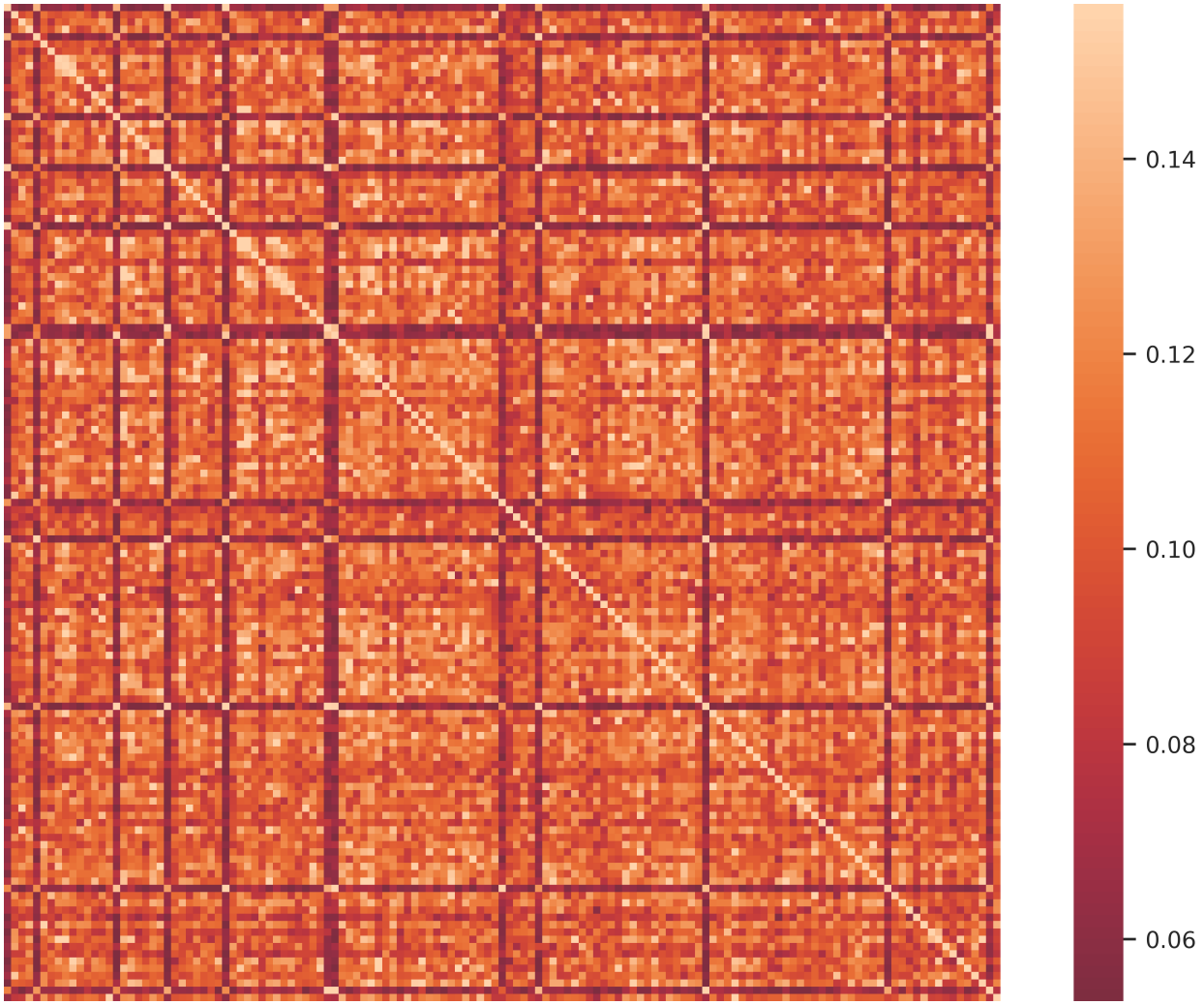


Figure 31. Average correlation coefficients between all series at the last forecasted time step for the `solar-10min` dataset. The bounds of the color map are set to the 0.02 and 0.98 quantiles of the data.

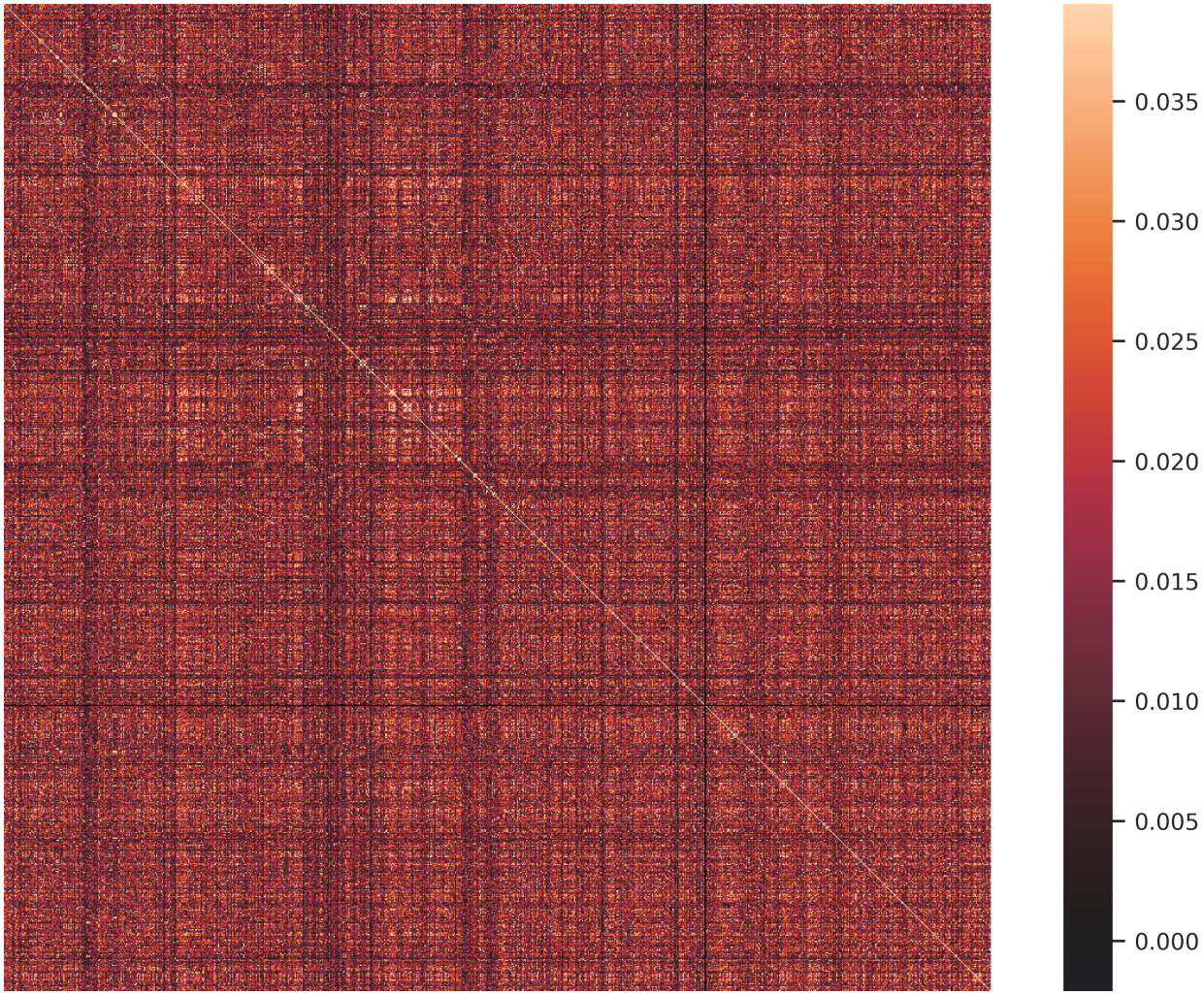


Figure 32. Average correlation coefficients between all series at the last forecasted time step for the `traffic` dataset. The bounds of the colour map are set to the 0.02 and 0.98 quantiles of the data. The black lines are caused by variables with constant forecasts, thus preventing them from having correlation coefficients.

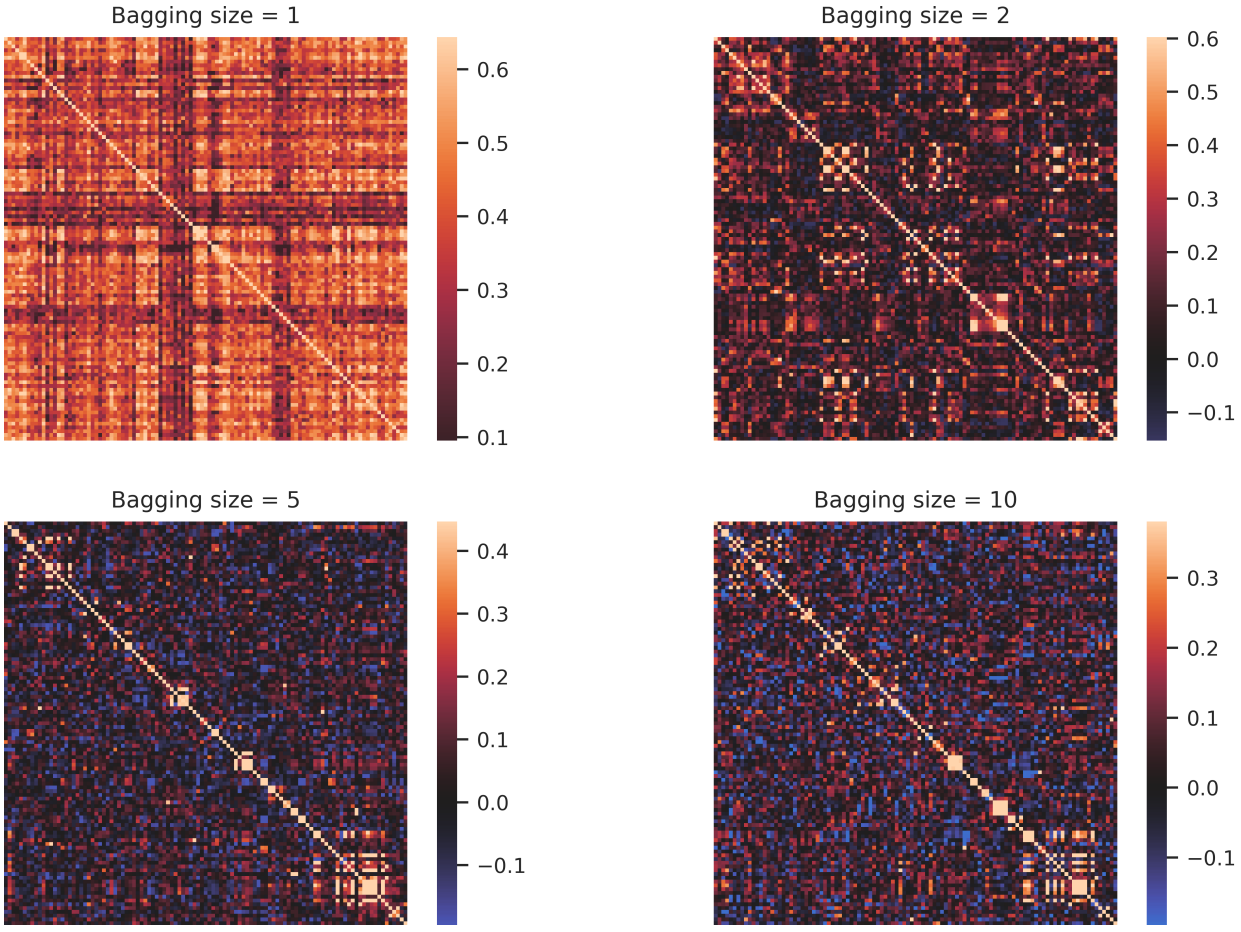


Figure 33. Average correlation coefficients between all series at the last forecasted time step for the `fred-md` dataset for various bagging sizes. The bounds of the color maps are set to the 0.02 and 0.98 quantiles of the data.