
High-Dimensional Multivariate Forecasting with Low-Rank Gaussian Copula Processes

David Salinas
NAVER LABS Europe *
david.salinas@naverlabs.com

Michael Bohlke-Schneider
Amazon Research
bohlkem@amazon.com

Laurent Callot
Amazon Research
lcallot@amazon.com

Roberto Medicò
Ghent University *
roberto.medico91@gmail.com

Jan Gasthaus
Amazon Research
gasthaus@amazon.com

Abstract

Predicting the dependencies between observations from multiple time series is critical for applications such as anomaly detection, financial risk management, causal analysis, or demand forecasting. However, the computational and numerical difficulties of estimating time-varying and high-dimensional covariance matrices often limits existing methods to handling at most a few hundred dimensions or requires making strong assumptions on the dependence between series. We propose to combine an RNN-based time series model with a Gaussian copula process output model with a low-rank covariance structure to reduce the computational complexity and handle non-Gaussian marginal distributions. This permits to drastically reduce the number of parameters and consequently allows the modeling of time-varying correlations of thousands of time series. We show on several real-world datasets that our method provides significant accuracy improvements over state-of-the-art baselines and perform an ablation study analyzing the contributions of the different components of our model.

1 Introduction

The goal of forecasting is to predict the distribution of future time series values. Forecasting tasks frequently require predicting several related time series, such as multiple metrics for a compute fleet or multiple products of the same category in demand forecasting. While these time series are often dependent, they are commonly assumed to be (conditionally) independent in high-dimensional settings because of the hurdle of estimating large covariance matrices.

Assuming independence, however, makes such methods unsuited for applications in which the correlations between time series play an important role. This is the case in finance, where risk minimizing portfolios cannot be constructed without a forecast of the covariance of assets. In retail, a method providing a probabilistic forecast for different sellers should take competition relationships and cannibalization effects into account. In anomaly detection, observing several nodes deviating from their expected behavior can be cause for alarm even if no single node exhibits clear signs of anomalous behavior.

Multivariate forecasting has been an important topic in the statistics and econometrics literature. Several multivariate extensions of classical univariate methods are widely used, such as vector autoregressions (VAR) extending autoregressive models [19], multivariate state-space models [7], or

*Work done while being at Amazon Research

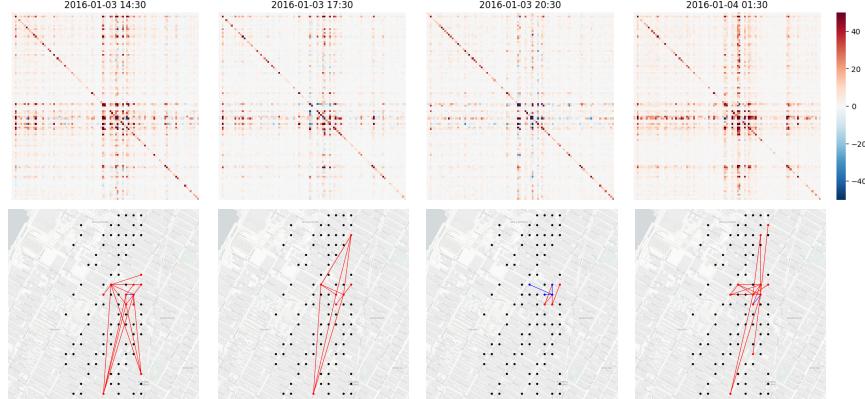


Figure 1: Top: covariance matrix predicted by our model for taxi traffic time series for 1214 locations in New-York at 4 different hours of a Sunday (only a neighborhood of 103 series is shown here, for clearer visualization). Bottom: Correlation graph obtained by keeping only pairs with covariance above a fixed threshold at the same hours. Both spatial and temporal relations are learned from the data as the covariance evolves over time and edges connect locations that are close to each other.

multivariate generalized autoregressive conditional heteroskedasticity (MGARCH) models [2]. The rapid increase in the difficulty of estimating these models due to the growth in number of parameters with the dimension of the problem have been binding constraints to move beyond low-dimensional cases. To alleviate these limitations, researchers have used dimensionality reduction methods and regularization, see for instance [3, 5] for VAR models and [34, 9] for MGARCH models, but these models remain unsuited for applications with more than a few hundreds dimensions [23].

Forecasting can be seen as an instance of sequence modeling, a topic which has been intensively studied by the machine learning community. Deep learning-based sequence models have been successfully applied to audio signals [33], language modeling [13, 30], and general density estimation of univariate sequences [22, 21]. Similar sequence modeling techniques have also been used in the context of forecasting to make probabilistic predictions for collections of real or integer-valued time series [26, 36, 16]. These approaches fit a global (i.e. shared) sequence-to-sequence model to a collection of time series, but generate statistically independent predictions. Outside the forecasting domain, similar methods have also been applied to (low-dimensional) multivariate dependent time series, e.g. two-dimensional time series of drawing trajectories [13, 14].

Two main issues prevent the estimation of high-dimensional multivariate time series models. The first one is the $O(N^2)$ scaling of the number of parameters required to express the covariance matrix where N denotes the dimension. Using dimensionality reduction techniques like PCA as a pre-processing step is a common approach to alleviate this problem, but it separates the estimation of the model from the preprocessing step, leading to decreased performance. This motivated [27] to perform such a factorization jointly with the model estimation. In this paper we show how the low rank plus diagonal covariance structure of the *factor analysis* model [29, 25, 10, 24] can be used in combination with Gaussian copula processes [37] to an LSTM-RNN [15] to jointly learn the temporal dynamics and the (time-varying) covariance structure, while significantly reducing the number of parameters that need to be estimated.

The second issue affects not only multivariate models, but all global time series models, i.e. models that estimate a single model for a collection of time series: In real-world data, the magnitudes of the time series can vary drastically between different series of the same data set, often spanning several orders of magnitude. In online retail demand forecasting, for example, item sales follow a power-law distribution, with a large number of items selling only a few units throughout the year, and a few popular items selling thousands of units per day [26]. The challenge posed by this for estimating global models across time series has been noted in previous work [26, 37, 27]. Several approaches have been proposed to alleviate this problem, including simple, fixed invertible transformations such as the square-root or logarithmic transformations, and the data-adaptive Box-Cox transform [4],

that aims to map a potentially heavy-tailed distribution to a Normal distribution. Other approaches includes removing scale with mean-scaling [26], or with a separate network [27].

Here, we propose to address this problem by modeling each time series' marginal distribution separately using a non-parametric estimate of its cumulative distribution function (CDF). Using this CDF estimate as the marginal transformation in a Gaussian copula (following [17, 18, 1]) effectively addresses the challenges posed by scaling, as it decouples the estimation of marginal distributions from the temporal dynamics and the dependency structure.

The work most closely related to ours is the recent work [32], which also proposes to combine deep autoregressive models with copula to model correlated time series. Their approach uses a nonparametric estimate of the copula, whereas we employ a Gaussian copula with low-rank structure that is learned jointly with the rest of the model. The nonparametric copula estimate requires splitting a N -dimensional cube into ε^{-N} many pieces (where N is the time series dimension and ε is a desired precision), making it difficult to scale that approach to large dimensions. The method also requires the marginal distributions and the dependency structure to be time-invariant, an assumption which is often violated in practice as shown in Fig. 1. A concurrent approach was proposed in [35] which also uses Copula and estimates marginal quantile functions with the approach proposed in [11] and models the Cholesky factor as the output of a neural network. Two important differences are that this approach requires to estimate $O(N^2)$ parameters to model the covariance matrix instead of $O(N)$ with the low-rank approach that we propose, another difference is the use of a non-parametric estimator for the marginal quantile functions.

The main contributions of this paper are:

- a method for probabilistic high-dimensional multivariate forecasting (scaling to dimensions up to an order of magnitude larger than previously reported in [23]),
- a parametrization of the output distribution based on a low-rank-plus-diagonal covariance matrix enabling this scaling by significantly reducing the number of parameters,
- a copula-based approach for handling different scales and modeling non-Gaussian data,
- an empirical study on artificial and six real-world datasets showing how this method improves accuracy over the state of the art while scaling to large dimensions.

The rest of the paper is structured as follows: In Section 2 we introduce the probabilistic forecasting problem and describe the overall structure of our model. We then describe how we can use the empirical marginal distributions in a Gaussian copula to address the scaling problem and handle non-Gaussian distributions in Section 3. In Section 4 we describe the parametrization of the covariance matrix with low-rank-plus-diagonal structure, and how the resulting model can be viewed as a low-rank Gaussian copula process. Finally, we report experiments with real-world datasets that demonstrate how these contributions combine to allow our model to generate correlated predictions that outperform state-of-the-art methods in terms of accuracy.

2 Autoregressive RNN Model for Probabilistic Multivariate Forecasting

Let us denote the values of a multivariate time series by $z_{i,t} \in \mathcal{D}$, where $i \in \{1, 2, \dots, N\}$ indexes the individual univariate component time series, and t indexes time. The domain \mathcal{D} is assumed to be either \mathbb{R} or \mathbb{N} . We will denote the multivariate observation vector at time t by $\mathbf{z}_t \in \mathcal{D}^N$. Given T observations $\mathbf{z}_1, \dots, \mathbf{z}_T$, we are interested in forecasting the future values for τ time units, i.e. we want to estimate the joint conditional distribution $P(\mathbf{z}_{T+1}, \dots, \mathbf{z}_{T+\tau} | \mathbf{z}_1, \dots, \mathbf{z}_T)$. In a nutshell, our model takes the form of a non-linear, deterministic state space model whose state $\mathbf{h}_{i,t} \in \mathbb{R}^k$ evolves independently for each time series i according to transition dynamics φ ,

$$\mathbf{h}_{i,t} = \varphi_{\theta_h}(\mathbf{h}_{i,t-1}, z_{i,t-1}) \quad i = 1, \dots, N, \quad (1)$$

where the transition dynamics φ are parametrized using a LSTM-RNN [15]. Note that the LSTM is unrolled for each time series separately, but parameters are tied across time series. Given the state values $\mathbf{h}_{i,t}$ for all time series $i = 1, 2, \dots, N$ and denoting by \mathbf{h}_t the collection of state values for all series at time t , we parametrize the joint emission distribution using a Gaussian copula,

$$p(\mathbf{z}_t | \mathbf{h}_t) = \mathcal{N}([f_1(z_{1,t}), f_2(z_{2,t}), \dots, f_N(z_{N,t})]^T | \boldsymbol{\mu}(\mathbf{h}_t), \Sigma(\mathbf{h}_t)). \quad (2)$$

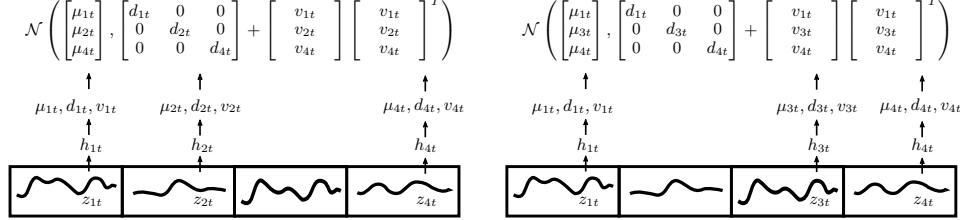


Figure 2: Illustration of our model parametrization. During training, dimensions are sampled at random and a local LSTM is unrolled on each of them individually (1, 2, 4, then 1, 3, 4 in the example). The parameters governing the state updates and parameter projections are shared for all time series. This parametrization can express the Low-rank Gaussian distribution on sets of series that varies during training or prediction.

The transformations $f_i : \mathcal{D} \rightarrow \mathbb{R}$ here are invertible mappings of the form $\Phi^{-1} \circ \hat{F}_i$, where Φ denotes the cumulative distribution function of the standard normal distribution, and \hat{F}_i denotes an estimate of the marginal distribution of the i -th time series $z_{i,1}, \dots, z_{i,T}$. The role of these functions f_i is to transform the data for the i -th time series such that marginally it follows a standard normal distribution. The functions $\mu(\cdot)$ and $\Sigma(\cdot)$ map the state \mathbf{h}_t to the mean and covariance of a Gaussian distribution over the transformed observations (described in more detail in Section 4).

Under this model, we can factorize the joint distribution of the observations as

$$p(\mathbf{z}_1, \dots, \mathbf{z}_{T+\tau}) = \prod_{t=1}^{T+\tau} p(\mathbf{z}_t | \mathbf{z}_1, \dots, \mathbf{z}_{t-1}) = \prod_{t=1}^{T+\tau} p(\mathbf{z}_t | \mathbf{h}_t). \quad (3)$$

Both the state update function φ and the mappings $\mu(\cdot)$ and $\Sigma(\cdot)$ have free parameters that are learned from the data. We denote θ the vector of all free parameters, consisting of the parameters of the state update θ_h as well as θ_μ and θ_Σ which denote the free parameters in $\mu(\cdot)$ and $\Sigma(\cdot)$. Given θ and \mathbf{h}_{T+1} , we can produce Monte Carlo samples from the joint predictive distribution

$$p(\mathbf{z}_{T+1}, \dots, \mathbf{z}_{T+\tau} | \mathbf{z}_1, \dots, \mathbf{z}_T) = p(\mathbf{z}_{T+1}, \dots, \mathbf{z}_{T+\tau} | \mathbf{h}_{T+1}) = \prod_{t=T+1}^{T+\tau} p(\mathbf{z}_t | \mathbf{h}_t) \quad (4)$$

by sequentially sampling from $p(\mathbf{z}_t | \mathbf{h}_t)$ and updating \mathbf{h}_t using φ ². We learn the parameters θ from the observed data $\mathbf{z}_1, \dots, \mathbf{z}_T$ using maximum likelihood estimation by i.e. by minimizing the loss function

$$-\log p(\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_T) = -\sum_{t=1}^T \log p(\mathbf{z}_t | \mathbf{h}_t), \quad (5)$$

using stochastic gradient descent-based optimization. To handle long time series, we employ a data augmentation strategy which randomly samples fixed-size slices of length $T' + \tau$ from the time series during training, where we fix the *context length* hyperparameter T' to τ . During prediction, only the last T' time steps are used in computing the initial state for prediction.

3 Gaussian Copula

A copula function $C : [0, 1]^N \rightarrow [0, 1]$ is the CDF of a joint distribution of a collection of real random variables U_1, \dots, U_N with uniform marginal distribution [8], i.e.

$$C(u_1, \dots, u_N) = P(U_1 \leq u_1, \dots, U_N \leq u_N).$$

Sklar's theorem [28] states that any joint cumulative distribution F admits a representation in terms of its univariate marginals F_i and a copula function C ,

$$F(z_1, \dots, z_N) = C(F_1(z_1), \dots, F_N(z_N)).$$

²Note that the model complexity scales linearly with the number of Monte Carlo samples.

When the marginals are continuous the copula C is unique and is given by the joint distribution of the *probability integral transforms* of the original variables, i.e. $\mathbf{u} \sim C$ where $u_i = F_i(z_i)$. Furthermore, if z_i is continuous then $u_i \sim \mathcal{U}(0, 1)$.

A common modeling choice for C is to use the Gaussian copula, defined by:

$$C(F_1(z_1), \dots, F_d(z_N)) = \phi_{\mu, \Sigma}(\Phi^{-1}(F_1(z_1)), \dots, \Phi^{-1}(F_d(z_N))),$$

where $\Phi : \mathbb{R} \rightarrow [0, 1]$ is the CDF of the standard normal and $\phi_{\mu, \Sigma}$ is a multivariate normal distribution parametrized with $\mu \in \mathbb{R}^N$ and $\Sigma \in \mathbb{R}^{N \times N}$. In this model, the observations \mathbf{z} , the marginally uniform random variables \mathbf{u} and the Gaussian random variables \mathbf{x} are related as follows:

$$\mathbf{x} \xrightarrow{\Phi} \mathbf{u} \xrightarrow{F^{-1}} \mathbf{z} \quad \mathbf{z} \xrightarrow{F} \mathbf{u} \xrightarrow{\Phi^{-1}} \mathbf{x}.$$

Setting $f_i = \Phi^{-1} \circ \hat{F}_i$ results in the model in Eq. (2).

The marginal distributions F_i are not given a priori and need to be estimated from data. We use the non-parametric approach of [17] proposed in the context of estimating high-dimensional distributions with sparse covariance structure. In particular, they use the empirical CDF of the marginal distributions,

$$\hat{F}_i(v) = \frac{1}{m} \sum_{t=1}^m \mathbb{1}_{z_{it} \leq v},$$

where m observations are considered. As we require the transformations f_i to be differentiable, we use a linearly-interpolated version of the empirical CDF resulting in a piecewise-constant derivative $\hat{F}'(\mathbf{u})$. This allow us to write the log-density of the original observations under our model as

$$\begin{aligned} \log p(\mathbf{z}; \mu, \Sigma) &= \log \phi_{\mu, \Sigma}(\Phi^{-1}(\hat{F}(\mathbf{z}))) + \log \frac{d}{d\mathbf{z}} \Phi^{-1}(\hat{F}(\mathbf{z})) \\ &= \log \phi_{\mu, \Sigma}(\Phi^{-1}(\hat{F}(\mathbf{z}))) + \log \frac{d}{d\mathbf{u}} \Phi^{-1}(\mathbf{u}) + \log \frac{d}{d\mathbf{z}} \hat{F}(\mathbf{z}) \\ &= \log \phi_{\mu, \Sigma}(\Phi^{-1}(\hat{F}(\mathbf{z}))) - \log \phi(\Phi^{-1}(\hat{F}(\mathbf{z}))) + \log \hat{F}'(\mathbf{z}) \end{aligned}$$

which are the individual terms in the total loss (5) where ϕ is the probability density function of the standard normal.

The number of past observations m used to estimate the empirical CDFs is an hyperparameter and left constant in our experiments with $m = 100$ ³.

4 Low-rank Gaussian Process Parametrization

After applying the marginal transformations $f_i(\cdot)$ our model assumes a joint multivariate Gaussian distribution over the transformed data. In this section we describe how the parameters $\mu(\mathbf{h}_t)$ and $\Sigma(\mathbf{h}_t)$ of this emission distribution are obtained from the LSTM state \mathbf{h}_t .

We begin by describing how a low-rank-plus-diagonal parametrization of the covariance matrix can be used to keep the computational complexity and the number of parameters manageable as the number of time series N grows. We then show how, by viewing the emission distribution as a time-varying low-rank Gaussian Process $g_t \sim \text{GP}(\tilde{\mu}_t(\cdot), k_t(\cdot, \cdot))$, we can train the model by only considering a subset of time series in each mini-batch further alleviating memory constraints and allowing the model to be applied to very high-dimensional sets of time series.

Let us denote the vector of transformed observations by

$$\mathbf{x}_t = f(\mathbf{z}_t) = [f_1(z_{1,t}), f_2(z_{2,t}), \dots, f_N(z_{N,t})]^T,$$

so that $p(\mathbf{x}_t | \mathbf{h}_t) = \mathcal{N}(\mathbf{x}_t | \mu(\mathbf{h}_t), \Sigma(\mathbf{h}_t))$. The covariance matrix $\Sigma(\mathbf{h}_t)$ is a $N \times N$ symmetric positive definite matrix with $O(N^2)$ free parameters. Evaluating the Gaussian likelihood naïvely

³This makes the underlying assumption that the marginal distributions are stationary, which is violated e.g. in case of a linear trend. Standard time series techniques such as de-trending or differencing can be used to pre-process the data such that this assumption is satisfied.

requires $O(N^3)$ operations. Using a structured parametrization of the covariance matrix as the sum of a diagonal matrix and a low rank matrix, $\Sigma = D + VV^T$ where $D \in \mathbb{R}^{N \times N}$ is diagonal and $V \in \mathbb{R}^{N \times r}$, results in a compact representation with $O(N \times r)$ parameters. This allows the likelihood to be evaluated using $O(Nr^2 + r^3)$ operations. As the rank hyperparameter r can typically be chosen to be much smaller than N , this leads to a significant speedup. In all our low-rank experiments we use $r = 10$. We investigate the sensitivity to this parameter of accuracy and speed in the Appendix.

Recall from Eq. 1 that $\mathbf{h}_{i,t}$ represents the state of an LSTM unrolled with values preceding $z_{i,t}$. In order to define the mapping $\Sigma(\mathbf{h}_t)$, we define mappings for its components

$$\Sigma(\mathbf{h}_t) = \begin{bmatrix} d_1(\mathbf{h}_{1,t}) & 0 \\ \ddots & \ddots \\ 0 & d_N(\mathbf{h}_{N,t}) \end{bmatrix} + \begin{bmatrix} \mathbf{v}_1(\mathbf{h}_{1,t}) \\ \vdots \\ \mathbf{v}_N(\mathbf{h}_{N,t}) \end{bmatrix} \begin{bmatrix} \mathbf{v}_1(\mathbf{h}_{1,t}) \\ \vdots \\ \mathbf{v}_N(\mathbf{h}_{N,t}) \end{bmatrix}^T = D_t + V_t V_t^T.$$

Note that the component mappings d_i and \mathbf{v}_i depend only on the state $\mathbf{h}_{i,t}$ for the i -th component time series, but not on the states of the other time series. Instead of learning separate mappings d_i , \mathbf{v}_i , and μ_i for each time series, we parametrize them in terms of the shared functions \tilde{d} , $\tilde{\mathbf{v}}$, and $\tilde{\mu}$, respectively. These functions depend on an E -dimensional feature vector $\mathbf{e}_i \in \mathbb{R}^E$ for each individual time series. The vectors \mathbf{e}_i can either be features of the time series that are known a priori, or can be embeddings that are learned with the rest of the model (or a combination of both).

Define the vector $\mathbf{y}_{i,t} = [\mathbf{h}_{i,t}; \mathbf{e}_i]^T \in \mathbb{R}^{p \times 1}$, which concatenates the state for time series i at time t with the features \mathbf{e}_i of the i -th time series and use the following parametrization:

$$\begin{aligned} \mu_i(\mathbf{h}_{i,t}) &= \tilde{\mu}(\mathbf{y}_{i,t}) = \mathbf{w}_\mu^T \mathbf{y}_{i,t} \\ d_i(\mathbf{h}_{i,t}) &= \tilde{d}(\mathbf{y}_{i,t}) = s(\mathbf{w}_d^T \mathbf{y}_{i,t}) \\ \mathbf{v}_i(\mathbf{h}_{i,t}) &= \tilde{\mathbf{v}}(\mathbf{y}_{i,t}) = W_v \mathbf{y}_{i,t}, \end{aligned}$$

where $s(x) = \log(1 + e^x)$ maps to positive values, $\mathbf{w}_\mu \in \mathbb{R}^{p \times 1}$, $\mathbf{w}_d \in \mathbb{R}^{p \times 1}$, $W_v \in \mathbb{R}^{r \times p}$ are parameters.

All parameters $\theta_\mu = \{\mathbf{w}_\mu, \mathbf{w}_{\tilde{\mu}}\}$, $\theta_\Sigma = \{\mathbf{w}_d, W_v, \mathbf{w}_{\tilde{d}}\}$ as well as the LSTM update parameters θ_h are learned by optimizing Eq. 5. These parameters are shared for all time series and can therefore be used to parametrize a GP. We can view the distribution of \mathbf{x}_t as a Gaussian process evaluated at points $\mathbf{y}_{i,t}$, i.e. $x_{i,t} = g_t(\mathbf{y}_{i,t})$, where $g_t \sim \text{GP}(\tilde{\mu}(\cdot), k(\cdot, \cdot))$, with $k(\mathbf{y}, \mathbf{y}') = \mathbb{1}_{\mathbf{y}=\mathbf{y}'} \tilde{d}(\mathbf{y}) + \tilde{\mathbf{v}}(\mathbf{y})^T \tilde{\mathbf{v}}(\mathbf{y}')$. Using this view it becomes apparent that we can train the model by evaluating the Gaussian terms in the loss only on random subsets of the time series in each iteration, i.e. we can train the model using batches of size $B \ll N$ as illustrated in Figure 2 (in our experiments we use $B = 20$). Further, if prior information about the covariance structure is available (e.g. in the case of spatial data the covariance might be directly related to the distance between points), this information can be easily incorporated directly into the kernel, either by exclusively using a pre-specified kernel or by combining it with the learned, time-varying kernel specified above.

5 Experiments

Synthetic experiment. We first perform an experiment on synthetic data demonstrating that our approach can recover complex time-varying low-rank covariance patterns from multi-dimensional observations. An artificial dataset is generated by drawing T observations from a normal distribution with time-varying mean and covariance matrix, $\mathbf{z}_t \sim \mathcal{N}(\rho_t \mathbf{u}, \Sigma_t)$ where $\rho_t = \sin(t)$, $\Sigma_t = U S_t U^T$ and

$$S_t = \begin{bmatrix} \sigma_1^2 & \rho_t \sigma_1 \sigma_2 \\ \rho_t \sigma_1 \sigma_2 & \sigma_2^2 \end{bmatrix}$$

The coefficients of $\mathbf{u} \in \mathbb{R}^{N \times 1}$ and $U \in \mathbb{R}^{N \times r}$ are drawn uniformly in $[a, b]$ and σ_1, σ_2 are fixed constants. By construction, the rank of Σ_t is 2. Both the mean and correlation coefficient of the two underlying latent variables oscillate through time as ρ_t oscillates between -1 and 1. In our experiments, the constants are set to $\sigma_1 = \sigma_2 = 0.1$, $a = -0.5$, $b = 0.5$ and $T = 24,000$.

In Figure 3, we compare the one-step-ahead predicted covariance given by our model, i.e. the lower triangle of $\Sigma(\mathbf{h}_t)$, to the true covariance, showing that the model is able to recover the complex underlying pattern of the covariance matrix.

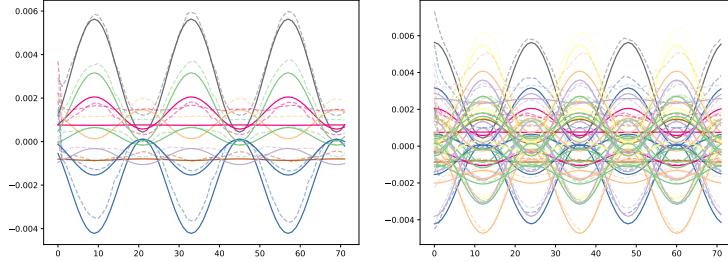


Figure 3: True (solid line) and predicted (dashed line) covariance after training with $N = 4$ (left) and $N = 8$ (right) time series. Each line corresponds to an entry in the lower triangle of Σ_t (including the diagonal, i.e. 10 lines in the left plot, 28 in the right).

Experiments with real-world datasets. The following publicly-available datasets are used to compare the accuracy of different multivariate forecasting models.

- Exchange rate: daily exchange rate between 8 currencies as used in [16]
- Solar: hourly photo-voltaic production of 137 stations in Alabama State used in [16]
- Electricity: hourly time series of the electricity consumption of 370 customers [6]
- Traffic: hourly occupancy rate, between 0 and 1, of 963 San Francisco car lanes [6]
- Taxi: spatio-temporal traffic time series of New York taxi rides [31] taken at 1214 locations every 30 minutes in the months of January 2015 (training set) and January 2016 (test set)
- Wikipedia: daily page views of 2000 Wikipedia pages used in [11]

Each dataset is split into a training and test set by using all data prior to a fixed date for the training and by using rolling windows for the test set. We measure accuracy on forecasts starting on time points equally spaced after the last point seen for training. For hourly datasets, accuracy is measured on 7 rolling time windows, for all other datasets we use 5 time windows, except for taxi, where 57 windows are used in order to cover the full test set. The number of steps predicted τ , domain, time-frequency, dimension N and time-steps available for training T is given in the appendix for all datasets.

Evaluation against baseline and ablation study. As we are looking into modeling correlated time series, only methods that are able to produce correlated samples are considered in our comparisons. The first baseline is VAR, a multivariate linear vector auto-regressive model using lag 1 and a lag corresponding to the periodicity of the data. The second is GARCH, a multivariate conditional heteroskedasticity model proposed by [34] with implementation from [12]. More details about these methods can be found in the supplement.

We also compare with different RNN architectures, distributions, and data transformation schemes to show the benefit of the low-rank Gaussian Copula Process that we propose. The most straightforward alternative to our approach is a single global LSTM that receives and predicts all target dimensions at once. We refer to this architecture as Vec-LSTM. We compare this architecture with the GP approach described in Section 4, where the LSTM is unrolled on each dimensions separately before reconstructing the joint distribution. For the output distribution in the Vec-LSTM architecture, we compare independent⁴, low-rank and full-rank normal distributions. For the data transformation we compare the copula approach that we propose, the mean scaling operation proposed in [26], and no transformation.

⁴Note that samples are still correlated with a diagonal noise due to the conditioning on the LSTM state.

baseline	architecture	data transformation	distribution	CRPS ratio	CRPS-Sum ratio	num params ratio
VAR	-	-	-	10.0	10.9	35.0
GARCH	-	-	-	7.8	6.3	6.2
Vec-LSTM-ind	Vec-LSTM	None	Independent	3.6	6.8	13.9
Vec-LSTM-ind-scaling	Vec-LSTM	Mean-scaling	Independent	1.4	1.4	13.9
Vec-LSTM-fullrank	Vec-LSTM	None	Full-rank	29.1	44.4	103.4
Vec-LSTM-fullrank-scaling	Vec-LSTM	Mean-scaling	Full-rank	22.5	37.6	103.4
Vec-LSTM-lowrank-Copula	Vec-LSTM	Copula	Low-rank	1.1	1.7	20.3
GP	GP	None	Low-rank	4.5	9.5	1.0
GP-scaling	GP	Mean-scaling	Low-rank	2.0	3.4	1.0
GP-Copula	GP	Copula	Low-rank	1.0	1.0	1.0

Table 1: Baselines summary and average ratio compared to GP-Copula for CRPS, CRPS-Sum and number of parameters on all datasets.

The description of the baselines as well as their average performance compared to GP-Copula are given in Table 1. For evaluation, we generate 400 samples from each model and evaluate multi-step accuracy using the continuous ranked probability score metric [20] that measures the accuracy of the predicted distribution (see supplement for details). We compute the CRPS metric on each time series individually (CRPS) as well on the sum of all time series (CRPS-Sum). Both metrics are averaged over the prediction horizon and over the evaluation time points. RNN models are trained only once with the dates preceding the first rolling time point and the same trained model is then used on all rolling evaluations.

Table 2 reports the CRPS-Sum accuracy of all methods (some entries are missing due to models requiring too much memory or having divergent losses). The individual time series CRPS as well as mean squared error are also reported in the supplement. Models that do not have data transformations are generally less accurate and more unstable. We believe this to be caused by the large scale variation between series also noted in [26, 27]. In particular, the copula transformation performs better than mean-scaling for GP, where GP-Copula significantly outperforms GP-scaling.

The GP-Copula model that we propose provides significant accuracy improvements on most datasets. In our comparison CRPS and CRPS-Sum are improved by on average 10% and 40% (respectively) compared to the second best models for those metrics Vec-LSTM-lowrank-Copula and Vec-LSTM-ind-scaling. One factor might be that the training is made more robust by adding randomness, as GP models need to predict different groups of series for each training example, making it harder to overfit. Note also that the number of parameters is drastically smaller compared to Vec-LSTM architectures. For the traffic dataset, the GP models have 44K parameters to estimate compared to 1.1M in a Vec-LSTM with a low-rank distribution and 38M parameters with a full-rank distribution. The complexity of the number of parameters are also given in Table 3.

We also qualitatively assess the covariance structure predicted by our model. In Fig. 1, we plot the predicted correlation matrix for several time steps after training on the Taxi dataset. We follow the approach in [17] and reconstruct the covariance graph by truncating edges whose correlation coefficient is less than a threshold kept constant over time. Fig. 1 shows the spatio-temporal correlation graph obtained at different hours. The predicted correlation matrices show how the model reconstructs the evolving topology of spatial relationships in the city traffic. Covariance matrices predicted over time by our model can also be found in the appendix for other datasets.

Additional details concerning the processing of the datasets, hyper-parameter optimization, evaluations, and model are given in the supplement. The code to perform the evaluations of our methods and different baselines is available at https://github.com/mbohlkeschneider/gluonts/tree/mv_release.

6 Conclusion

We presented an approach to obtain probabilistic forecast of high-dimensional multivariate time series. By using a low-rank approximation, we can avoid the potentially very large number of parameters of a full covariate matrix and by using a low-rank Gaussian Copula process we can stably optimize directly parameters of an autoregressive model. We believe that such techniques allowing to estimate high-dimensional time varying covariance matrices may open the door to several applications in anomaly detection, imputation or graph analysis for time series data.

dataset estimator	CRPS-Sum					
	exchange	solar	elec	traffic	taxi	wiki
VAR	0.010+/-0.000	0.524+/-0.001	0.031+/-0.000	0.144+/-0.000	0.292+/-0.000	3.400+/-0.003
GARCH	0.020+/-0.000	0.869+/-0.000	0.278+/-0.000	0.368+/-0.000	-	-
Vec-LSTM-ind	0.009+/-0.000	0.470+/-0.039	0.731+/-0.007	0.110+/-0.020	0.429+/-0.000	0.801+/-0.029
Vec-LSTM-ind-scaling	0.008+/-0.001	0.391+/-0.017	0.025+/-0.001	0.087+/-0.041	0.506+/-0.005	0.133+/-0.002
Vec-LSTM-fullrank	0.646+/-0.114	0.956+/-0.000	0.999+/-0.000	-	-	-
Vec-LSTM-fullrank-scaling	0.394+/-0.174	0.920+/-0.035	0.747+/-0.020	-	-	-
Vec-LSTM-lowrank-Copula	0.007+/-0.000	0.319+/-0.011	0.064+/-0.008	0.103+/-0.006	0.326+/-0.007	0.241+/-0.033
GP	0.011+/-0.001	0.828+/-0.010	0.947+/-0.016	2.198+/-0.774	0.425+/-0.199	0.933+/-0.003
GP-scaling	0.009+/-0.000	0.368+/-0.012	0.022+/-0.000	0.079+/-0.000	0.183+/-0.395	1.483+/-1.034
GP-Copula	0.007+/-0.000	0.337+/-0.024	0.024+/-0.002	0.078+/-0.002	0.208+/-0.183	0.086+/-0.004

Table 2: CRPS-sum accuracy comparison (lower is better, best two methods are in bold). Mean and std are obtained by rerunning each method three times.

	input	output	
	independent	low-rank	full-rank
Vec-LSTM	$O(Nk)$	$O(Nk)$	$O(Nrk)$
GP	$O(k)$	$O(k)$	$O(rk)$

Table 3: Number of parameters for input and output projection of different models. We recall that N and k denotes the dimension and size of the LSTM state.

References

- [1] Wolfgang Aussenegg and Christian Cech. A new copula approach for high-dimensional real world portfolios. *University of Applied Sciences bfi Vienna, Austria. Working paper series*, 68 (2012):1–26, 2012.
- [2] Luc Bauwens, Sébastien Laurent, and Jeroen VK Rombouts. Multivariate garch models: a survey. *Journal of applied econometrics*, 21(1):79–109, 2006.
- [3] Ben S Bernanke, Jean Boivin, and Piotr Eliasz. Measuring the effects of monetary policy: a factor-augmented vector autoregressive (favar) approach. *The Quarterly journal of economics*, 120(1):387–422, 2005.
- [4] G. E. P. Box and D. R. Cox. An analysis of transformations. *Journal of the Royal Statistical Society. Series B (Methodological)*, 26(2):211–252, 1964.
- [5] Laurent AF Callot and Anders B Kock. Oracle efficient estimation and forecasting with the adaptive lasso and the adaptive group lasso in vector autoregressions. *Essays in Nonlinear Time Series Econometrics*, pages 238–268, 2014.
- [6] Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository, 2017. URL <http://archive.ics.uci.edu/ml>.
- [7] James Durbin and Siem Jan Koopman. *Time series analysis by state space methods*, volume 38. Oxford University Press, 2012.
- [8] Gal Elidan. Copulas in machine learning. In Piotr Jaworski, Fabrizio Durante, and Wolfgang Karl Härdle, editors, *Copulae in Mathematical and Quantitative Finance*, pages 39–60, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-35407-6.
- [9] Robert Engle. Dynamic conditional correlation: A simple class of multivariate generalized autoregressive conditional heteroskedasticity models. *Journal of Business & Economic Statistics*, 20(3):339–350, 2002.
- [10] B S Everitt. *An Introduction to Latent Variable Models*. Chapman and Hill, 1984.
- [11] Jan Gasthaus, Benidis Benidis, Konstantinos, Yuyang Wang, Syama S. Rangapuram, David Salinas, Valentin Flunkert, and Tim Januschowski. Probabilistic forecasting with spline quantile function RNNs. *AISTATS*, 2019.

- [12] Alexios Ghalanos. *rmgarch: Multivariate GARCH models.*, 2019. R package version 1.3-6.
- [13] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.
- [14] Karol Gregor, Ivo Danihelka, Alex Graves, and Daan Wierstra. DRAW: A recurrent neural network for image generation. *CoRR*, abs/1502.04623, 2015. URL <http://arxiv.org/abs/1502.04623>.
- [15] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [16] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long- and short-term temporal patterns with deep neural networks. *CoRR*, abs/1703.07015, 2017. URL <http://arxiv.org/abs/1703.07015>.
- [17] Han Liu, John Lafferty, and Larry Wasserman. The nonparanormal: Semiparametric estimation of high dimensional undirected graphs. *Journal of Machine Learning Research*, 10(Oct):2295–2328, 2009.
- [18] Han Liu, Fang Han, Ming Yuan, John Lafferty, Larry Wasserman, et al. High-dimensional semiparametric Gaussian copula graphical models. *The Annals of Statistics*, 40(4):2293–2326, 2012.
- [19] Helmut Lütkepohl. *New introduction to multiple time series analysis*. Springer Science & Business Media, 2005.
- [20] James E Matheson and Robert L Winkler. Scoring rules for continuous probability distributions. *Management science*, 22(10):1087–1096, 1976.
- [21] Junier Oliva, Avinava Dubey, Manzil Zaheer, Barnabas Poczos, Ruslan Salakhutdinov, Eric Xing, and Jeff Schneider. Transformation autoregressive networks. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3898–3907, Stockholm, Sweden, 10–15 Jul 2018. PMLR.
- [22] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 2338–2347. Curran Associates, Inc., 2017.
- [23] Andrew J Patton. A review of copula models for economic time series. *Journal of Multivariate Analysis*, 110:4–18, 2012.
- [24] Sam Roweis and Zoubin Ghahramani. A unifying review of linear Gaussian models. *Neural Computation*, 11(2):305–345, 1999.
- [25] Donald B Rubin and Dorothy T Thayer. EM algorithms for ML factor analysis. *Psychometrika*, 47(1):69–76, 1982.
- [26] David Salinas, Valentin Flunkert, and Jan Gasthaus. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *CoRR*, abs/1704.04110, 2017. URL <http://arxiv.org/abs/1704.04110>.
- [27] Rajat Sen, Hsiang-Fu Yu, and Inderjit Dhillon. Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting. *arXiv preprint arXiv:1905.03806*, 2019.
- [28] M Sklar. Fonctions de répartition à n dimensions et leurs marges. *Publications de l’Institut de Statistique de l’Université de Paris*, 8:229–231, 1959.

- [29] C. Spearman. "general intelligence," objectively determined and measured. *The American Journal of Psychology*, 15(2):201–292, 1904. ISSN 00029556. URL <http://www.jstor.org/stable/1412107>.
- [30] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112, 2014.
- [31] NYC Taxi and Limousine Commission. TLC trip record data. <https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page>, 2015.
- [32] J. Toubeau, J. Bottieau, F. Vallée, and Z. De Grève. Deep learning-based multivariate probabilistic forecasting for short-term scheduling in power markets. *IEEE Transactions on Power Systems*, 34(2):1203–1215, March 2019. ISSN 0885-8950. doi: 10.1109/TPWRS.2018.2870041.
- [33] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *CoRR*, abs/1609.03499, 2016.
- [34] Roy Van der Weide. Go-garch: a multivariate generalized orthogonal garch model. *Journal of Applied Econometrics*, 17(5):549–564, 2002.
- [35] Ruofeng Wen and Kari Torkkola. Deep generative quantile-copula models for probabilistic forecasting. *arXiv preprint arXiv:1907.10697*, 2019.
- [36] Ruofeng Wen, Kari Torkkola, and Balakrishnan Narayanaswamy. A multi-horizon quantile recurrent forecaster. *arXiv preprint arXiv:1711.11053*, 2017.
- [37] Andrew Gordon Wilson and Zoubin Ghahramani. Copula processes. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 2*, NIPS’10, pages 2460–2468, USA, 2010. Curran Associates Inc. URL <http://dl.acm.org/citation.cfm?id=2997046.2997170>.

High-Dimensional Multivariate Forecasting with Low-Rank Gaussian Copula Processes

Supplementary material

David Salinas NAVER LABS Europe * david.salinas@naverlabs.com	Michael Bohlke-Schneider Amazon Research bohlkem@amazon.com	Laurent Callot Amazon Research lcallot@amazon.com
Roberto Medicò Ghent University * roberto.medico91@gmail.com	Jan Gasthaus Amazon Research gasthaus@amazon.com	

Contents

A Multivariate Likelihood	2
B Empirical CDF	2
C Effect of rank on low-rank approximation	3
D Baseline additional description	3
E Hyperparameter optimization	4
F Dataset details	5
G Metrics	5
G.1 Continuous Ranked Probability Score (CRPS)	5
G.2 Mean Squared Error (MSE)	6
H Comparison with forecasting methods with diagonal covariance	6
I Predicted correlation matrices	8
J Effect of the number of evaluation samples on CRPS and inference runtime	8
K Additional experiments details	8
L Open-source implementation of our model	10

*Work done while being at Amazon Research

A Multivariate Likelihood

The probability density function of a multivariate normal distribution can be expressed as

$$\phi_{\mu, \Sigma}(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^d |L|}} \exp\left(-\frac{1}{2} \|L^{-1}(\mathbf{x} - \mu)\|^2\right)$$

where $\mu \in \mathbb{R}^d$ and $L \in \mathbb{R}^{d \times d}$ is the Cholesky factor of the covariance matrix $\Sigma = LL^T$. This form is particularly amenable to computation using common neural network frameworks, as we only need to compute the determinant of a triangular matrix and solve a triangular system, which can both be done in $O(d^2)$.

This approach works in modest dimensions, but the quadratic cost in computational time and number of parameters becomes prohibitive when considering larger dimensions. This issue can be avoided by utilizing a low-rank matrix $\Sigma = D + VV^T$, where $D \in \mathbb{R}^{d \times d}$ and diagonal, and $V \in \mathbb{R}^{d \times r}$ where $r \ll d$ is a rank hyper-parameter. This parametrization of the covariance matrix also arises from the *factor analysis* model [18, 13, 3, 12], i.e. as the marginal distribution of \mathbf{x} under the latent variable model

$$\mathbf{y} \sim \mathcal{N}(0, I), \quad x \sim \mathcal{N}(V\mathbf{x}, D)$$

When the covariance matrix is restricted in this way, it has $O(dr)$ parameters, and the Gaussian likelihood can be computed in $O(dr^2 + r^3)$ time.

In particular, the log-likelihood $\log \phi_{\mu, \Sigma}(x)$ can be evaluated by first computing an r -by- r matrix $C = I_r + V^T D^{-1} V$ in $O(dr^2)$ time, followed by computing its Cholesky decomposition $C = L_C L_C^T$ in $O(r^3)$ time. Using the matrix determinant lemma in [7] we can write

$$\begin{aligned} \log |\Sigma| &= \log |D + VV^T| \\ &= \log |C| + \log |D| \\ &= 2 \log |L_C| + \log |D| \end{aligned}$$

which can be computed in $O(d + r)$ as L_C and D are triangular. Given L_C , the Mahalanobis distance $\mathbf{x}^T \Sigma^{-1} \mathbf{x}$ can also be computed efficiently. By the Woodbury matrix identity we have $\Sigma^{-1} = D^{-1} - D^{-1} V C^{-1} V^T D^{-1}$. We can then write,

$$\begin{aligned} \mathbf{x}^T \Sigma^{-1} \mathbf{x} &= \mathbf{x}^T (D^{-1} - D^{-1} V C^{-1} V^T D^{-1}) \mathbf{x} \\ &= \mathbf{x}^T D^{-1} \mathbf{x} - \mathbf{x}^T (D^{-1} V C^{-1} V^T D^{-1}) \mathbf{x} \\ &= \mathbf{x}^T D^{-1} \mathbf{x} - \mathbf{y}^T C^{-1} \mathbf{y}, \text{ with } \mathbf{y} = V^T D^{-1} \mathbf{x} \\ &= \mathbf{x}^T D^{-1} \mathbf{x} - \|L_C^{-1} \mathbf{y}\|^2 \end{aligned}$$

The first term of the final equality can be computed in $O(d)$ and the second term can be computed with back-substitution in $O(r^2)$, so that the total time is $O(dr^2 + r^3)$ and the number of parameters is $O(dr)$.

The factor analysis latent variable model is closely related to PCA [21]: If we restrict the diagonal matrix D to a multiple of the identity matrix, $D = \psi I$, we obtain a probabilistic version of PCA, from which the classical PCA can be recovered in the limit $\psi \rightarrow 0$. Previous work [19] has applied PCA as a preprocessing step for uncovering latent structure. Here we propose an end-to-end approach that learns the structure of the covariance matrix jointly with the time series model.

B Empirical CDF

The naive empirical CDF estimator can exhibit large variance and the following truncated estimator from [10] is used instead:

$$\tilde{F}_i(v) = \begin{cases} \delta_m & \text{if } \hat{F}_i(v) < \delta_m \\ \hat{F}_i(v) & \text{if } \delta_m \leq \hat{F}_i(v) \leq 1 - \delta_m \\ 1 - \delta_m & \text{if } \hat{F}_i(v) > 1 - \delta_m \end{cases}$$

where choosing $\delta_m = \frac{1}{4m^{1/4}\sqrt{\pi \log m}}$ strikes the right bias-variance trade-off [10].

Further, we add jitter noise at training when computing the mapping f_i to smooth the CDF for discrete data.

C Effect of rank on low-rank approximation

The effect of the low-rank approximation is analyzed in Figure 1 and Table 1 on the electricity dataset. As expected, the negative log-likelihood training loss decreases as the rank r of V increases in Figure 1. Table 1 shows the impact of the rank on the training/test loss. While the training loss decreases as the rank is increased, our model reaches its best performance on the test dataset with rank values 32/64. For higher ranks (128 and 256), the difference between training and test loss increases. This indicates that the high rank models may over-fit to the training data due to the flexibility of high-rank covariance matrix approximation.

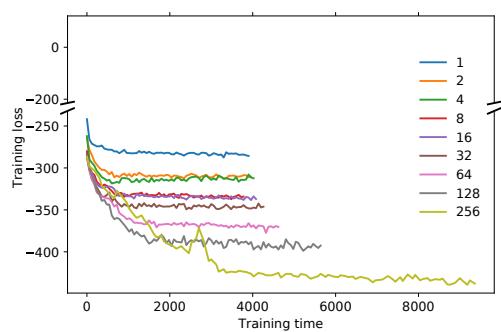


Figure 1: Training loss vs training time when increasing rank on the electricity dataset.

rank	test NLL	train NLL
1	-291.4+/-8.2	-288.9+/-8.2
2	-306.2+/-6.7	-304.8+/-5.7
4	-319.3+/-4.9	-312.1+/-3.5
8	-333.6+/-7.7	-330.2+/-6.3
16	-334.8+/-4.9	-337.5+/-4.
32	-341.8+/-6.8	-345.2+/-17.0
64	-338.5+/-10.9	-360.5+/-10.7
128	-326.6+/-20.1	-393.7+/-26.1
256	-238.0+/-38.4	-423.1+/-20.7

Table 1: Error metrics when evaluating on the electricity dataset with increasing rank. We show the mean +/- 95% confidence interval over five runs.

D Baseline additional description

The GARCH [22] (Generalize Orthogonal - Generalize Autoregressive Conditional Heteroskedasticity) model is a composit model providing dynamics for the conditional mean and conditional covariance matrix of a multivariate system. The model for the conditional mean is, here, an autoregressive model of order one,

$$z_{i,t} = \alpha_i + \beta_i z_{i,t-1} + \epsilon_{i,t}.$$

Define $\epsilon_t = [\epsilon_{1,t}, \dots, \epsilon_{N,t}]'$. To predict the conditional covariance matrix, the GARCH model maps ϵ_t to a set of $F = \min(N, T)$ independent factors $\mathbf{f}_t = [f_{1,t}, \dots, f_{F,t}]'$, $\epsilon_t = A\mathbf{f}_t$, where A is a time independent, invertible matrix of dimension $[N \times F]$. The conditional variance $\sigma_{j,t}^2$, $j = [1, \dots, F]$ of each of the factors is modeled using independent GARCH-type models, here a GARCH(1,0):

$$\sigma_{j,t}^2 = \omega_j + \gamma_j \sigma_{j,t-1}^2.$$

The linear mapping A and the factors \mathbf{f} are estimated using the ICA method of [2, 24]. Let $\mathbf{f}_t = H_t^{1/2} \mathbf{x}_t$, where \mathbf{x}_t is a vector of independent random variables with conditional mean zero and conditional variance one. H_t is the diagonal matrix of conditional variances of the factors with diagonal $[\sigma_{1,t}^2, \dots, \sigma_{F,t}^2]$. The conditional covariance matrix \mathbf{z}_t is then given by $\Sigma_t = A' H_t A$. We use the GARCH implementation of [5].

The VAR model is a multivariate linear vector auto-regression using lag 1 and a lag l where l corresponds to the periodicity of the data,

$$\mathbf{z}_t = \boldsymbol{\mu} + B_1 \mathbf{z}_{t-1} + B_l \mathbf{z}_{t-l} + \epsilon_t.$$

$\boldsymbol{\mu}$ is a vector of intercepts of dimension $[N \times 1]$, and B_1 and B_l are parameter matrices of dimension $[N \times N]$. Letting $z_i = [z_{i,l+1}, \dots, z_{i,T}]'$, $\mathbf{x}_t = [\mathbf{z}'_{t-1}, \mathbf{z}'_{t-l}]'$, and $X = [\mathbf{x}_{l+1}, \dots, \mathbf{x}_T]'$, each individual equation of the model can be written in stacked form as

$$z_i = \mu_i + X\boldsymbol{\theta}_i + \epsilon_i, \quad (1)$$

where μ_i is a scalar intercept parameter and $\boldsymbol{\theta}_i$ is a parameter vector of length $2N$.

The parameters of equation 1 are estimated by the Lasso implemented in [4] using the procedure described in [8]. The estimated parameters are the minimizers of the loss function

$$L(\mu_i, \boldsymbol{\theta}_i) = \frac{1}{T-l} \|z_i - \mu_i - X\boldsymbol{\theta}_i\|_{\ell_2}^2 + 2\lambda_i \|\boldsymbol{\theta}_i\|_{\ell_1},$$

where $\|\cdot\|_{\ell_2}$ is the ℓ_2 -norm and $\|\cdot\|_{\ell_1}$ ℓ_1 -norm. λ_i is a tuning parameter whose selection procedure is explained below.

E Hyperparameter optimization

Parameters are learned with SGD using ADAM optimizer with batch of 16 elements, 12 regularization with 1e-8 and gradient clipped to 10.0. For all methods, we apply 10000 gradient updates in total and decay the learning rate by 2 after 500 consecutive updates without improvement.

Table 2 lists the parameters that are tuned as well as the value of hyper-parameters that are not tuned and kept constant across all datasets.

HYPERPARAMETER	VALUE OR RANGE SEARCHED
learning rate	[1e-4, 1e-4, 1e-2]
LSTM cells	[10, 20, 40]
LSTM layers	2
rank	10
num eval samples	400
conditioning length m	100
sampling dimension B	20
dropout	0.01
batch size	16

Table 2: Hyper-parameters values fixed or range searched in hyper-parameter tuning.

To tune hyper-parameters of RNN methods we perform a grid-search of 12 parameters on Electricity and Exchange for Vec-LSTM-ind-scaling. The best hyperparameter for a method is set as the hyperparameter having the best average rank for CRPS. The best learning-rate/number-cells found for Vec-LSTM-ind-scaling is 1e-3 / 40, as LSTM and GP baselines has many variations, we use the same hyperparameter for all variants.

The Lasso estimator of the VAR model has a single Hyperparameter λ_i for each equation. The best value of the parameter is selected within the sequence of values considered by the path-wise coordinate descent algorithm [4]. λ_i is in the range $[\lambda_{i,min}, \lambda_{i,max}]$, where $\lambda_{i,max}$ is the smallest value of λ_i such that all penalized parameters of the VAR are set to zero while $\lambda_{i,min} = \varepsilon \lambda_{i,max}$ where $\varepsilon = 0.0001$ if $N < T$ and $\varepsilon = 0.01$ otherwise[4]. The best value of λ_i the value in the sequence that minimizes a Bayesian Information Criterion [8].

For the GARCH model, we performed a search among all combinations of mean and variance model specifications. The mean models considered were: AR(1), AR(Seasonal), VAR(1), VAR(Seasonal). The models for the variance components considered were: GARCH(1,0), GARCH(1,1), fGARCH(1,0) and fGARCH(1,1) [5]. We found that the only specification able to consistently converge in even the smallest of our datasets was using AR(1) as the mean model and GARCH(1,0) for the variance components.

F Dataset details

dataset	τ (num steps predicted)	domain	frequency	dimension N	time steps T
Exchange rate	30	\mathbb{R}^+	daily	8	6071
Solar	24	\mathbb{R}^+	hourly	137	7009
Electricity	24	\mathbb{R}^+	hourly	370	5790
Traffic	24	\mathbb{R}^+	hourly	963	10413
Taxi	24	\mathbb{N}	30-min	1214	1488
Wikipedia	30	\mathbb{N}	daily	2000	792

Table 3: Summary of the datasets used to test the models. Number of steps forecasted, data domain \mathcal{D} , frequency of observations, dimension of series N , and number of time steps T .

Datasets (or their processing) will be made available after publication. Table 3 shows the properties of the used datasets. We only describe the processing for Taxi as all other datasets have been used in previous publications. The dataset obtained from [20] is preprocessed with the following steps similarly to [17]:

- Data cleaning: removal of outliers in terms of average speed (> 45.31 mph), trip duration (> 720 minutes), trip distance (> 23 miles) and trip fare (> 86.6);
- Data reduction: the dataset is reduced to the most active areas by retaining the area bounded by (40.70,-74.07) and (40.84,-73.95), expressed as (latitude, longitude) pairs;
- Data binning: first, the data is binned over time, using a frequency of 30 minutes; afterwards, the data is aggregated spatially, by binning latitude and longitude on a grid with spatial granularity of 0.001;
- For each subregion in the spatial grid and within each 30 minutes interval, the number of pickups and dropoffs are summed;
- Data filtering: the least active areas are filtered out from the data, by retaining only areas with at least 80% non-zero observations. This results in a total of 1214 time series.

We use January 2015 for the training set and January 2016 for the test set as in [17].

G Metrics

G.1 Continuous Ranked Probability Score (CRPS)

The *continuous ranked probability score* (CRPS) [11, 6] measures the compatibility of a probability distribution F (represented by its quantile function F^{-1}) with an observation y . The *pinball loss* (or *quantile loss*) at a quantile level $\alpha \in [0, 1]$ and with a predicted α -th quantile q is defined as

$$\Lambda_\alpha(q, y) = (\alpha - \mathcal{I}_{[y < q]})(y - q). \quad (2)$$

The CRPS has an intuitive definition as the pinball loss integrated over all quantile levels $\alpha \in [0, 1]$,

$$\text{CRPS}(F^{-1}, y) = \int_0^1 2\Lambda_\alpha(F^{-1}(\alpha), y) d\alpha. \quad (3)$$

An important property of the CRPS is that it is a *proper scoring rule* [6], implying that the CRPS is minimized when the predictive distribution is equal to the distribution from which the data is drawn.

In our setting, we are interested in evaluating the accuracy of the prediction compared to an observation $\mathbf{z}_t \in \mathbb{R}^N$. To do so, we generate predictions in the form of 400 samples which allows to estimate the quantile function F^{-1} predicted by the model.

We then report average marginal CRPS over dimensions and over predicted steps in Table 4, e.g. we report

$$\mathbb{E}_{i,t}[\text{CRPS}(F_i^{-1}, z_{i,t})]$$

where F_i^{-1} is obtained by sorting the samples drawn when predicting $z_{i,t}$.

CRPS						
dataset estimator	exchange	solar	elec	traffic	taxi	wiki
VAR	0.015+/-0.000	0.595+/-0.000	0.060+/-0.000	0.222+/-0.000	0.410+/-0.000	4.101+/-0.002
GARCH	0.024+/-0.000	0.928+/-0.000	0.291+/-0.000	0.426+/-0.000	-	-
Vec-LSTM-ind	0.020+/-0.001	0.480+/-0.031	0.765+/-0.005	0.234+/-0.007	0.495+/-0.002	0.800+/-0.028
Vec-LSTM-ind-scaling	0.013+/-0.000	0.434+/-0.012	0.059+/-0.001	0.168+/-0.037	0.586+/-0.004	0.379+/-0.004
Vec-LSTM-fullrank	0.610+/-0.096	0.939+/-0.001	0.997+/-0.000	-	-	-
Vec-LSTM-fullrank-scaling	0.377+/-0.115	1.003+/-0.021	0.749+/-0.020	-	-	-
Vec-LSTM-lowrank-Copula	0.009+/-0.000	0.384+/-0.010	0.084+/-0.006	0.165+/-0.004	0.416+/-0.004	0.247+/-0.001
GP	0.029+/-0.000	0.834+/-0.002	0.900+/-0.023	1.255+/-0.562	0.475+/-0.177	0.870+/-0.011
GP-scaling	0.017+/-0.000	0.415+/-0.009	0.053+/-0.000	0.140+/-0.002	0.346+/-0.348	1.549+/-1.017
GP-Copula	0.008+/-0.000	0.371+/-0.022	0.056+/-0.002	0.133+/-0.001	0.360+/-0.201	0.236+/-0.000

Table 4: CRPS accuracy metrics (lower is better, best two methods are in bold). Mean and standard error are reported by running each method 3 times.

estimator	MSE	MSE-sum	num_params
VAR	-	-	-
GARCH	-	-	-
Vec-LSTM-ind	17.0	52.3	13.6
Vec-LSTM-ind-scaling	1.3	1.3	13.6
Vec-LSTM-fullrank	801.8	1545.6	103.4
Vec-LSTM-fullrank-scaling	985.3	1937.5	103.4
Vec-LSTM-lowrank-Copula	4.7	10.4	19.2
GP	122.3	1080.7	1.0
GP-scaling	1.1	3.0	1.0
GP-Copula	1.0	1.0	1.0

Table 5: Baselines summary and average ratio compared to GP-Copula for MSE, MSE-Sum, and number of parameters on all datasets.

To account for joint effect, we also report CRPS-Sum where accuracy is measured on the predicted distribution of the sum, e.g.

$$\mathbb{E}_t[\text{CRPS}(F^{-1}, \sum_i z_{i,t})]$$

Where F^{-1} is obtained by first summing samples across dimensions and then sorting to get quantiles.

Integrals are estimated with 10 equally-spaced quantiles.

G.2 Mean Squared Error (MSE)

The MSE is defined as the mean squared error over all time series, i.e., $i = 1, \dots, N$, and over the whole prediction range, i.e., $t = T - t_0 + 1, \dots, T$:

$$\text{MSE} = \frac{1}{N(T - t_0)} \sum_{i,t} (z_{i,t} - \hat{z}_{i,t})^2 \quad (4)$$

where z is the target and \hat{z} the predicted distribution mean. Tables 5 - 7 show the MSE results for the marginal MSE and the MSE-sum. The definition of MSE-sum is analogous to CRPS-sum.

H Comparison with forecasting methods with diagonal covariance

We evaluated our approach against DeepAR [14] and MQCNN [23], which we believe are a fair representation of the state-of-the-art in deep-learning-based forecasting. We also compared with DeepGLO [16] on two datasets provided by the authors. Table 8 lists the results of this comparison. Note that none of these competing approaches models correlations across time series in their forecasts (DeepGLO only provides point forecasts).

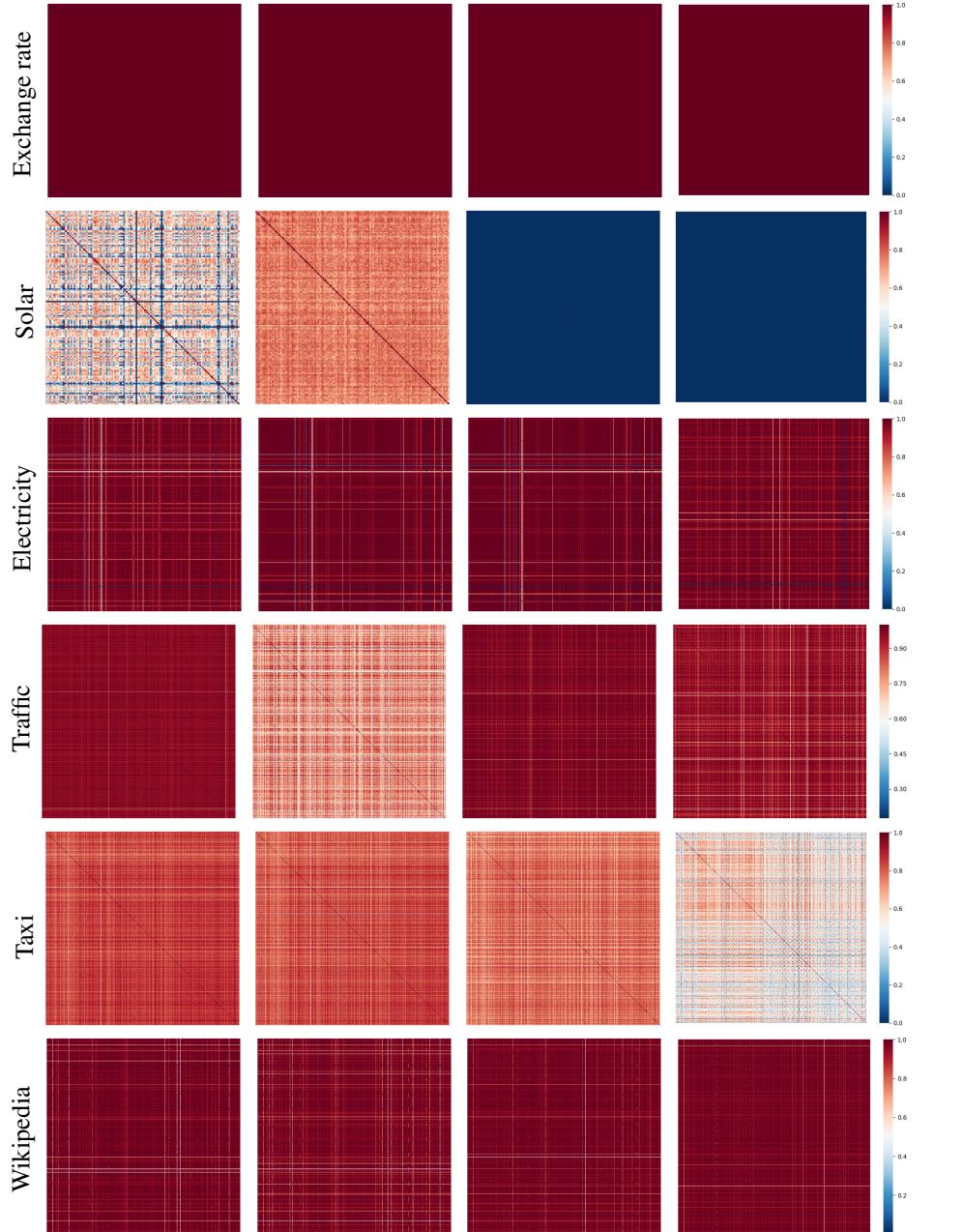


Figure 2: Correlation matrix predicted by our model at four different equally spaced time-steps with $step = 6$ for all datasets in this study. Exchange rate is nearly homoscedastic and most correlations are close to 1, because all currencies are relative to US dollar and therefore highly correlated. The remaining datasets are clearly heteroscedastic. For example, solar has low correlation at night and electricity/traffic/taxi follow day-night cycles. Wikipedia also shows heteroscedasticity across time.

MSE						
dataset estimator	exchange	solar	elec	traffic	taxi	wiki
VAR	4.4e-24/-2.2e-5	7.0e3+/-2.5e1	1.2e74/-5.4e3	5.1e-3+/-2.9e-6	-	-
GARCH	4.0e-2+/-5.3e-5	3.5e3+/-2.0e1	1.2e6+/-2.5e4	3.3e-3+/-1.8e-6	-	-
Vec-LSTM-ind	3.9e-44/-2.0e-4	9.9e2+/-2.8e2	2.6e74/-4.6e4	6.5e-4+/-1.1e-4	5.2e1+/-2.2e-1	5.2e7+/-3.8e5
Vec-LSTM-ind-scaling	1.6e-44/-2.6e-5	9.3e2+/-1.9e2	2.1e5+/-1.2e4	6.3e-4+/-5.6e-5	7.3e1+/-1.1e0	7.2e7+/-2.1e6
Vec-LSTM-fullrank	5.2e-1+/-1.5e-1	3.8e3+/-1.8e1	2.7e74/-2.3e2	-	-	-
Vec-LSTM-fullrank-scaling	6.5e-1+/-4.3e-2	3.8e3+/-6.9e1	3.2e74/-1.1e7	-	-	-
Vec-LSTM-lowrank-Copula	1.9e-44/-1.3e-6	2.9e3+/-1.1e2	5.5e6+/-1.2e6	1.5e-3+/-2.5e-6	5.1e1+/-3.2e-1	3.8e7+/-1.5e5
GP	3.0e-44/-4.8e-5	3.7e3+/-5.7e1	2.7e74/-2.0e3	5.1e-1+/-2.5e-1	5.9e1+/-2.0e1	5.4e7+/-2.3e4
GP-scaling	2.9e-44/-3.5e-5	1.1e3+/-3.3e1	1.8e5+/-1.4e4	5.2e-4+/-4.4e-6	2.7e1+/-1.0e1	5.5e7+/-3.6e7
GP-Copula	1.7e-44/-1.6e-5	9.8e2+/-5.2e1	2.4e5+/-5.5e4	6.9e-4+/-2.2e-5	3.1e1+/-1.4e0	4.0e7+/-1.6e9

Table 6: MSE accuracy metrics (lower is better). Mean and standard error are reported by running each method 3 times.

MSE-sum						
dataset estimator	exchange	solar	elec	traffic	taxi	wiki
VAR	1.2e0+/-8.6e-4	1.1e8+/-3.9e5	1.8e10+/-1.3e7	2.5e3+/-3.4e0	-	-
GARCH	1.1e0+/-2.0e-3	5.6e7+/-3.2e5	2.7e9+/-3.3e7	1.1e3+/-2.1e0	-	-
Vec-LSTM-ind	1.3e-2+/-7.0e-3	1.1e7+/-4.6e6	5.3e10+/-9.2e8	1.2e2+/-8.1e1	2.7e7+/-2.8e5	2.6e13+/-1.5e12
Vec-LSTM-ind-scaling	3.2e-3+/-1.3e-3	1.1e7+/-2.4e6	1.2e8+/-7.9e6	5.5e1+/-2.8e1	4.0e7+/-1.0e6	1.2e12+/-9.7e10
Vec-LSTM-fullrank	2.3e1+/-8.0e0	5.8e7+/-3.0e5	8.9e10+/-7.9e6	-	-	-
Vec-LSTM-fullrank-scaling	3.0e1+/-2.5e0	5.6e7+/-7.8e5	8.8e10+/-1.7e10	-	-	-
Vec-LSTM-lowrank-Copula	4.6e-3+/-8.3e-5	4.2e7+/-2.0e6	8.1e9+/-9.0e8	7.0e2+/-6.4e0	2.5e7+/-2.8e5	5.8e11+/-6.5e10
GP	7.2e-3+/-2.4e-3	5.5e7+/-1.0e6	8.6e10+/-1.9e9	4.3e5+/-2.2e5	3.3e7+/-1.8e7	3.5e13+/-9.5e10
GP-scaling	7.3e-3+/-1.8e-3	1.2e7+/-6.4e5	1.4e8+/-1.9e7	7.0e1+/-3.4e0	7.9e6+/-8.9e6	2.7e13+/-4.5e13
GP-Copula	4.2e-3+/-6.5e-4	1.2e7+/-8.3e5	1.5e8+/-3.5e7	6.2e1+/-4.3e0	1.0e7+/-1.1e6	1.9e12+/-2.2e15

Table 7: MSE-sum accuracy metrics (lower is better). Mean and standard error are reported by running each method 3 times.

dataset	exchange	solar	elec	traffic	taxi	wiki
DeepAR [14]	0.007	0.379	0.063	0.147	0.332	0.337
MQCNN [23]	0.013	0.482	0.078	0.177	0.657	0.277
GP-Copula (Ours)	0.008	0.371	0.056	0.133	0.360	0.236
dataset			electricity	traffic		
DeepGLO [16]			0.109	0.221		
TRMF [16]			0.105	0.210		
GP-Copula (Ours)			0.083	0.168		

Table 8: CRPS for additional baselines (left) and comparison with [16] when measuring WAPE (right).

I Predicted correlation matrices

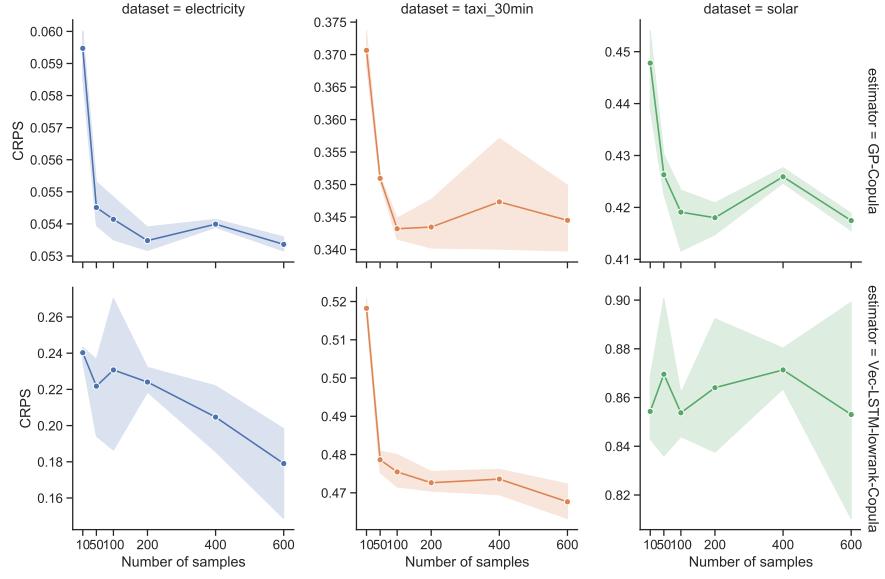
We illustrate the learned correlations of our model on all datasets in Figure 2.

J Effect of the number of evaluation samples on CRPS and inference runtime

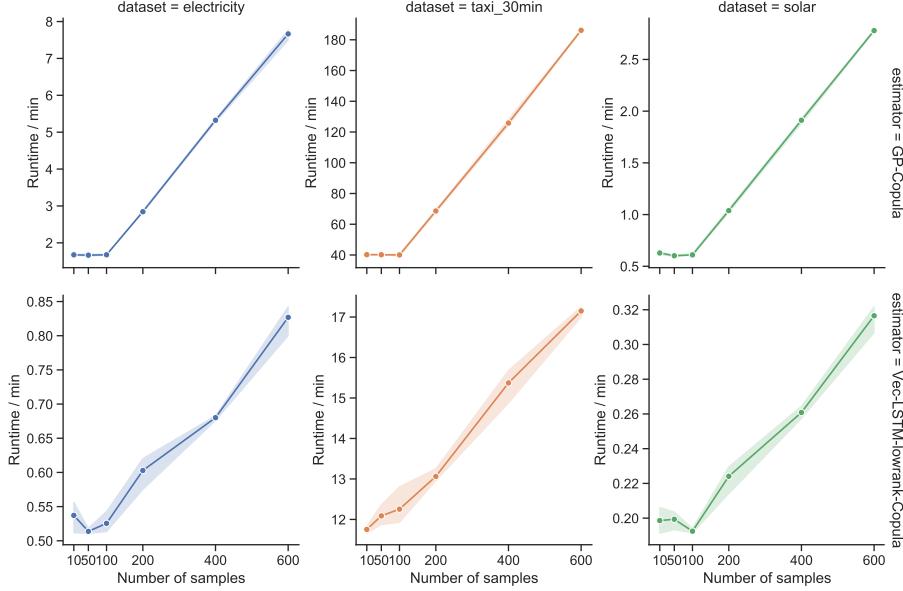
Figure 3 shows the effect of the number of evaluation samples on the CRPS and inference runtime. Drawing more than 100 samples only has a small effect on the CRPS and linearly increases the inference runtime.

K Additional experiments details

We use generic features to represent time. For hourly dataset, we use hour of day, day of week, day of month features. For daily dataset, we use day of week feature. For minutes granularity, we use minute of hour, hour of day and day of week features. All features are encoded with one number, for instance hour of day feature takes values in $[0, 23]$. Feature values are concatenated to the LSTM



(a) Effect of the number of evaluation samples (`num eval samples`) on the CRPS for electricity, solar, and taxi datasets. The line shows the mean CRPS over three independent runs and the shaded area shows the 95% confidence interval. Increasing the number of samples from 100 to 600 has a small effect on the CRPS (average CRPS over all datasets decreases from 0.272 to 0.271 for GP-Copula and from 0.52 to 0.50 for Vec-LSTM-lowrank-Copula, respectively).



(b) Effect of the number of evaluation samples (`num eval samples`) on the inference runtime. The inference runtime scales linearly with the number of drawn samples. The inferences runtimes for 10, 50, and 100 samples are similar due to initialization overhead. Note that the samples are only drawn during inference. Thus, the `num eval samples` parameter does not affect training runtime.

Figure 3: Effect of the number of evaluation samples on CRPS and inference runtime.

CRPS-sum						
dataset estimator	exchange	solar	elec	traffic	taxi	wiki
GP-Copula	0.007+-0.000	0.337+-0.024	0.024+-0.002	0.078+-0.002	0.208+-0.183	0.086+-0.004
GP-Copula (Gluonts)	0.007+-0.000	0.404+-0.009	0.027+-0.001	0.050+-0.003	0.159+-0.001	0.055+-0.005

Table 9: CRPS-sum accuracy metrics for the GluonTS implementation of our model (lower is better). Mean and standard error are reported by running each method 3 times.

CRPS						
dataset estimator	exchange	solar	elec	traffic	taxi	wiki
GP-Copula	0.008+-0.000	0.371+-0.022	0.056+-0.002	0.133+-0.001	0.360+-0.201	0.236+-0.000
GP-Copula (Gluonts)	0.009+-0.000	0.416+-0.007	0.054+-0.000	0.106+-0.002	0.339+-0.001	0.244+-0.003

Table 10: CRPS accuracy metrics for the GluonTS implementation of our model (lower is better). Mean and standard error are reported by running each method 3 times.

input at each time-step. We also lags values as input \mathcal{L} according to the time-frequency, [1, 24, 168] for hourly data, [1, 7, 14] for daily, and [1, 2, 4, 12, 24, 48] for 30 minutes data.

All models are evaluated on a Amazon Web Services c5.4xlarge instance with 16 cores and 32GB RAM. All RNNs models take under five hours to perform training and evaluation. Missing numbers in Table 4 happens either because Out-of-memory prevents training or NaNs appear during training because of unstable models. Finally, RNNs are combined with Zone-out regularization [9] and residual connections and MXNet is used as the neural network framework [15].

L Open-source implementation of our model

We re-implemented the model described in this paper in GluonTS [1], an open-source time series toolkit. To ensure re-reproducibility, we released a static version of the code online that is not part of the latest GluonTS releases (for which we cannot guarantee reproducibility over time) at https://github.com/mbohlkeschneider/gluon-ts/tree/mv_release. Tables 9 and 10 show the benchmark results of our re-implementation. Our GluonTS implementation performs similar to the implementation that was used in this paper. In the new implementation, we set the sampling dimension B to 2. Furthermore, we found that the piecewise-constant derivatives did not improve the results and removed them from our implementation.

References

- [1] Alexander Alexandrov, Konstantinos Benidis, Michael Bohlke-Schneider, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Danielle C Maddix, Syama Rangapuram, David Salinas, and Jasper Schulz. Gluonts: Probabilistic time series models in python. *arXiv preprint arXiv:1906.05264*, 2019.
- [2] Simon A Broda and Marc S Paoletta. Chicago: A fast and accurate method for portfolio risk calculation. *Journal of Financial Econometrics*, 7(4):412–436, 2009.
- [3] B S Everitt. *An Introduction to Latent Variable Models*. Chapman and Hill, 1984.
- [4] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010. URL <http://www.jstatsoft.org/v33/i01/>.
- [5] Alexios Ghalanos. *rmgarch: Multivariate GARCH models.*, 2019. R package version 1.3-6.
- [6] Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007.
- [7] David A Harville. Matrix algebra from a statistician’s perspective, 1998.
- [8] Anders Bredahl Kock and Laurent Callot. Oracle inequalities for high dimensional vector autoregressions. *Journal of Econometrics*, 186(2):325–344, 2015.
- [9] David Krueger, Tegan Maharaj, János Kramár, Mohammad Pezeshki, Nicolas Ballas, Nan Rosemary Ke, Anirudh Goyal, Yoshua Bengio, Hugo Larochelle, Aaron C. Courville, and Chris Pal. Zoneout: Regularizing rnns by randomly preserving hidden activations. *CoRR*, abs/1606.01305, 2016. URL <http://arxiv.org/abs/1606.01305>.
- [10] Han Liu, John Lafferty, and Larry Wasserman. The Nonparanormal: Semiparametric Estimation of High Dimensional Undirected Graphs. 10:2295–2328, 2009. ISSN 1532-4435. doi: 10.1016/0006-291X(91)91267-G. URL <http://arxiv.org/abs/0903.0649>.
- [11] James E Matheson and Robert L Winkler. Scoring rules for continuous probability distributions. *Management science*, 22(10):1087–1096, 1976.
- [12] Sam Roweis and Zoubin Ghahramani. A unifying review of linear Gaussian models. *Neural Computation*, 11(2):305–345, 1999.
- [13] Donald B Rubin and Dorothy T Thayer. EM algorithms for ML factor analysis. *Psychometrika*, 47(1):69–76, 1982.
- [14] David Salinas, Valentin Flunkert, and Jan Gasthaus. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *CoRR*, abs/1704.04110, 2017. URL <http://arxiv.org/abs/1704.04110>.
- [15] Matthias W. Seeger, Asmus Hetzel, Zhenwen Dai, and Neil D. Lawrence. Auto-differentiating linear algebra. *CoRR*, abs/1710.08717, 2017. URL <http://arxiv.org/abs/1710.08717>.
- [16] Rajat Sen, Hsiang-Fu Yu, and Inderjit Dhillon. Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting. *arXiv preprint arXiv:1905.03806*, 2019.
- [17] Gaurav Sharma. Taxi demand prediction new york city. <https://github.com/gauravtheP/Taxi-Demand-Prediction-New-York-City>, 2018.
- [18] C. Spearman. "general intelligence," objectively determined and measured. *The American Journal of Psychology*, 15(2):201–292, 1904. ISSN 00029556. URL <http://www.jstor.org/stable/1412107>.

- [19] James H Stock and Mark W Watson. Dynamic factor models, factor-augmented vector autoregressions, and structural vector autoregressions in macroeconomics. In *Handbook of macroeconomics*, volume 2, pages 415–525. Elsevier, 2016.
- [20] NYC Taxi and Limousine Commission. TLC trip record data. <https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page>, 2015.
- [21] Michael E Tipping and Christopher M Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999.
- [22] Roy Van der Weide. Go-garch: a multivariate generalized orthogonal garch model. *Journal of Applied Econometrics*, 17(5):549–564, 2002.
- [23] Ruofeng Wen, Kari Torkkola, and Balakrishnan Narayanaswamy. A multi-horizon quantile recurrent forecaster. *arXiv preprint arXiv:1711.11053*, 2017.
- [24] Kun Zhang and Laiwan Chan. Efficient factor garch models and factor-dcc models. *Quantitative Finance*, 9(1):71–91, 2009.