Normalizing Kalman Filters for Multivariate Time Series Analysis

Emmanuel de Bézenac[1 †, *] **Syama Sundar Rangapuram**[2 *], **Konstantinos Benidis**[2], **Michael Bohlke-Schneider**[2], **Richard Kurle**[3 †] **Lorenzo Stella**[2], **Hilaf Hasson**[2], **Patrick Gallinari**[1], **Tim Januschowski**[2]
[1]Sorbonne Université, [2]AWS AI Labs, [3]Technical University of Munich

Correspondence to: `emmanuel.de-bezenac@lip6.fr, rangapur@amazon.de`

Abstract

This paper tackles the modelling of large, complex and multivariate time series panels in a probabilistic setting. To this extent, we present a novel approach reconciling classical state space models with deep learning methods. By *augmenting* state space models with normalizing flows, we mitigate imprecisions stemming from idealized assumptions in state space models. The resulting model is highly flexible while still retaining many of the attractive properties of state space models, e.g., uncertainty and observation errors are properly accounted for, inference is tractable, sampling is efficient and good generalization performance is observed, even in low data regimes. We demonstrate competitiveness against state-of-the-art deep learning methods on the tasks of forecasting real world data and handling varying levels of missing data.

1 Introduction

In most real world applications of time series analysis, e.g., risk management in finance, cannibalization of products in retail or anomaly detection in cloud computing environments, time series are not mutually independent and an accurate modelling approach must take these dependencies into account [1]. The classical approach [2] is to extend standard univariate models resulting in vector autoregression [3], multivariate GARCH [4] and multivariate state space models [5, 6]. Although these approaches yield useful theoretical properties, they make idealized assumptions like Gaussianity, linear inter-dependencies, and are not scalable to even moderate number of time series [7] due to the number of parameters required to be estimated, which is restrictive for many modern applications involving large panels of time series. Recently, more expressive, scalable deep learning methods [8, 9] were developed for forecasting applications that learn a joint global model for multiple time series; however, they still assume that these time series are mutually independent.

In this paper we propose the *Normalizing Kalman Filter* (`NKF`), a novel approach for modelling and forecasting complex multivariate time series by augmenting classical linear Gaussian state space models (`LGM`) with normalizing flows [10]. The combined model allows us to leverage the flexibility of normalizing flows (NF), alleviating strong assumptions of traditional multivariate models, while still benefiting from the rich set of mathematical properties of `LGM`. In fact, we prove that despite modelling non-Gaussian data with nonlinear inter-dependencies, we can achieve exact inference since our model has closed-form expressions for filtering, smoothing and likelihood computation. We thus retain the main attractive properties of `LGM`, in contrast to related methods [11, 12]. Moreover, since

[*]Equal contribution.
[†]Work done while at Amazon.

our model is based on LGM, handling of missing data and integrating prior knowledge, e.g., seasonality and trend, becomes trivial. Therefore, the proposed model can be used in forecasting time series with missing or noisy data irrespective of whether the data regime is sparse (in terms of observed time points) or dense. More importantly, LGM directly gives us the ability to provide tractable multi-step ahead forecast distributions while accounting for all uncertainties; this is in contrast to recent deep learning-based autoregressive models [8, 1] that do not incorporate accumulated prediction errors into forecast distributions since predictions of the model are used as lagged inputs in a multi-step forecast scenario. For the forecasting application, we show that our method scales linearly with the number of dimensions and number of time points, unlike most of the existing work that exhibits quadratic scaling with the number of dimensions. The necessary structural assumptions do not result in a loss of generality or expressiveness of the overall model.

In summary, our main contributions are as follows:

- A tractable method for modelling non-Gaussian multivariate time series data with nonlinear inter-dependencies that has Kalman-like recursive updates for filtering and smoothing.
- A scalable, robust multivariate forecasting method that handles missing data naturally and provides tractable multi-step ahead forecast distributions while accounting for uncertainties unlike autoregressive models [8, 1].
- A thorough evaluation of applicability of normalizing flows in the context of high-dimensional time series forecasting for handling non-Gaussian multivariate data with nonlinear dependencies.

2 Normalizing Kalman Filters

Let $y_t \in \mathbb{R}^N$ denote the value of a multivariate time series at time t, with $y_{t,i} \in \mathbb{R}$ the value of the corresponding i-th univariate time series. Further, let $x_{t,i} \in \mathbb{R}^k$ be time varying covariate vectors associated to each univariate time series at time t, and $x_t := [x_{t,1}, \ldots, x_{t,N}] \in \mathbb{R}^{k \times N}$. Non-random and random variables are denoted by normal and bold letters, i.e., x and $\mathbf{x}$, respectively. We use the shorthand $y_{1:T}$ to denote the sequence $\{y_1, y_2, \ldots, y_T\}$.

2.1 Generative Model

The core assumption behind our Normalizing Kalman Filter (NKF) model is the existence of a latent state that evolves according to simple (linear) dynamics, with potentially complex and nonlinear dependencies between latent state and observations–and thus, among observations. More precisely, the dynamics of the latent state $\mathbf{l}_t \in \mathbb{R}^d$ are governed by a time-dependent *transition matrix* $F_t \in \mathbb{R}^{d \times d}$, up to additive Gaussian noise $\boldsymbol{\epsilon}_t$ as in (1b). The state is then mapped into the space of observations with *emission matrix* $A_t \in \mathbb{R}^{d \times N}$, and additive Gaussian noise $\boldsymbol{\varepsilon}_t$ before being transformed by a potentially nonlinear function $f_t : \mathbb{R}^N \to \mathbb{R}^N$ parametrized by Λ, generating observation $\mathbf{y}_t \in \mathbb{R}^N$:

$$\text{(NKF model)} \quad \begin{aligned} \mathbf{l}_1 &\sim \mathcal{N}(\mu_1, \Sigma_1) && && \text{(initial state)} && \text{(1a)} \\ \mathbf{l}_t &= F_t \mathbf{l}_{t-1} + \boldsymbol{\epsilon}_t, && \boldsymbol{\epsilon}_t \sim \mathcal{N}(0, \Sigma_t), && \text{(transition dynamics)} && \text{(1b)} \\ \mathbf{y}_t &= f_t(A_t^T \mathbf{l}_t + \boldsymbol{\varepsilon}_t), && \boldsymbol{\varepsilon}_t \sim \mathcal{N}(0, \Gamma_t). && \text{(observation model)} && \text{(1c)} \end{aligned}$$

With parameters Λ and $\Theta = (\mu_1, \Sigma_1, \{\Gamma_t, A_t\}_{t \geq 1}, \{\Sigma_t, F_t\}_{t \geq 2})$, the model is fully specified.[3] Note that the special case $f_t = \texttt{id}$ recovers the standard LGM where both the transition dynamics and the observation model are linear. In the following, this similarity with LGM will yield numerous computational benefits and it will further allow us to easily inject prior knowledge on the structural form of the dynamics (e.g., levels, trends, seasonalities [5]) for good generalization properties.

We consider a flexible nonlinear transformation for the observation model, assuming invertibility of f_t. This guarantees the conservation of probability mass and allows the evaluation of the associated density function at any given point of interest. In particular, the probability density of an observation

[3]Placing the noise before the non-linearity f_t in the observation model is important to obtain tractability for filtering and smoothing. However, this does not imply that data generated from a process where additive noise is added after the non-linear function cannot be modelled; in fact we use this particular model (nonlinear transformation with additive non-Gaussian noise) to generate data and test our method in the qualitative experiments in Section 4.1 (refer to appendix C.1 in the supplementary material for details).

y_t given the state l_t can be computed using the change of variables formula:

$$p(y_t|l_t;\Theta,\Lambda) = p_z(f_t^{-1}(y_t)|l_t;\Theta)\left|\det\left[\mathrm{Jac}_{y_t}(f_t^{-1})\right]\right|, \tag{2}$$

where the first term in $p_z(z_t|l_t;\Theta)$ is the density of the Gaussian variable $z_t := A_t l_t + \varepsilon_t$ conditioned on l_t, and the second is the absolute value of the determinant of the Jacobian of f_t^{-1} given Λ, evaluated at y_t. This equation and Figure 1 illustrate the intuition behind our approach: we would like f_t^{-1} to transform the observations such that the dynamics become simple and the noise is Gaussian.

Computing the density (2) raises several issues: (i) finding a flexible f_t while ensuring invertibility, (ii) being able to compute the inverse efficiently and (iii) tractability of the computation of the Jacobian term when the number of time series N is large. To this extent, we will take inspiration from *normalizing flows* [13, 14, 15], which are invertible neural networks that typically transform isotropic Gaussians to fit a more complex data distribution. These invertible networks are tailored to compute the Jacobian term efficiently. Moreover, they have proven to work very well for nonlinear high-dimensional data, e.g., images [14], both in terms of flexibility and generalization. In our approach, we apply these invertible neural networks to temporal data, using them to map the distribution p_z given by the `LGM` to the complex data distribution. This yields a powerful function f_t where the Jacobian term is computable in *linear time* in N.

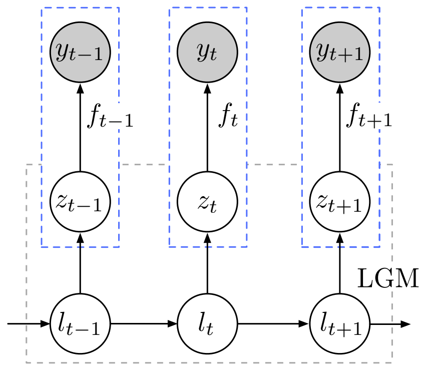


Figure 1: Generative model of the `NKF`. States l_t and *pseudo-observations* z_t are produced from an `LGM`, which are then transformed through a normalizing flow f_t, producing observations y_t.

Inference and Learning. With the presented generative model it is possible to do inference (e.g., filtering and smoothing) and training in a simple and tractable way. Similar to the `LGM`, computing the *filtered distribution* $p(l_t|y_{1:t};\Theta,\Lambda)$ is essential as it determines our current belief on the state having observed all the data up to time t, and it takes part in the computation of the likelihood of the model parameters as well as the forecast distribution. For general nonlinear state space models its computation is tedious as it involves integrating out previous states. Methods such as Particle Filters [16] resort to Monte Carlo approximations of these integrals but have difficulty scaling to high dimensions. Other methods circumvent this by locally linearizing the nonlinear transformation [17] or by using a finite sample approximation [18] in order to apply–in both cases– the techniques of the standard `LGM`, but introduce a bias. In contrast, our model allows for computing this quantity in a tractable and efficient manner, without resorting to any form of simplification or approximation. In fact, despite the nonlinear nature of f_t, the filtered distribution *remains* Gaussian and its parameters can be computed in closed form similarly to the Kalman Filter, as shown in the following proposition.

Proposition 1 (Filtering). The *filtered* distributions of the `NKF` model are *Gaussian* and are given by the filtered distributions of the corresponding `LGM` with pseudo-observations $z_t := f_t^{-1}(y_t),\ t \geq 1$. That is, $p(l_t|y_{1:t};\Theta,\Lambda) = p_{\mathtt{LGM}}(l_t|z_{1:t};\Theta)$ where $p_{\mathtt{LGM}}$ refers to the distribution given by the `LGM`.

This can be proved by induction using recursive Bayesian estimation and is available in Appendix A.1 along with the exact updates. Proposition 1 shows that filtered distributions for our model are available in closed-form and have the same computational complexity as that of the `LGM`, plus the complexity of the inverse of the nonlinear function f and the Jacobian term in (2).[4]

Our nonlinear model (1) is also amenable to smoothing, i.e., computing the smoothed posterior distribution $p(l_t|y_{1:T};\Theta,\Lambda)$, given past, present and future observations. Smoothing is a prevalent problem in many fields, with applications such as estimating the distribution of missing data and providing explanations in the context of offline anomaly detection. Smoothing can be obtained with a backward iterative approach using the quantities computed during a preliminary filtering pass, starting from the filtered distribution corresponding to step T, $p(l_T|y_{1:T};\Theta)$. Similar to filtering, smoothing updates also directly translate to the corresponding updates of the `LGM`.

[4]In Sec. 3, the complexity of the inverse is the same as the forward map, and the Jacobian term is linear in N.

Proposition 2 (Smoothing)**.** The *smoothed* distributions of the `NKF` model are *Gaussian* and are given by the smoothed distributions of the corresponding `LGM` with pseudo-observations $z_t := f_t^{-1}(y_t), t = 1, 2 \ldots, T$. That is, $p(l_t|y_{1:T}; \Theta, \Lambda) = p_{\texttt{LGM}}(l_t|z_{1:T}; \Theta)$.

The tractability of the computation of the filtered distribution implies that the likelihood of the model parameters can be computed efficiently by integrating out the latent state. In case of the standard `LGM`, the likelihood of its parameters Θ given the observations $z_{1:T}$ can be written as

$$\ell(\Theta) = \prod_{t=1}^{T} p_{\texttt{LGM}}(z_t|z_{1:t-1}; \Theta), \tag{3}$$

where $p_{\texttt{LGM}}(z_t|z_{1:t-1}; \Theta)$ is the predictive distribution of `LGM`. This distribution is available in closed-form thanks to the filtering updates [19]. We now show that the likelihood of our nonlinear model given the observations $\{y_{1:T}\}$ is essentially a reweighted version of this expression with $z_t = f_t^{-1}(y_t)$.

Proposition 3 (Likelihood)**.** The likelihood of the parameters (Θ, Λ) of the `NKF` model given the observations $\{y_{1:T}\}$ can be computed as

$$\ell(\Theta, \Lambda) = p(y_{1:T}; \Theta, \Lambda) = \prod_{t=1}^{T} p_{\texttt{LGM}}(z_t|z_{1:t-1}; \Theta) \left|\det \left[\mathrm{Jac}_{z_t}(f_t)\right]\right|^{-1}, \tag{4}$$

where $z_t = f_t^{-1}(y_t)$ and $p_{\texttt{LGM}}(z_t|z_{1:t-1}; \Theta)$ denotes the predictive distribution of `LGM`.

Hence, we can compute the likelihood of the parameters of the `NKF` exactly (Appendix A.3 contains the closed-form expressions), and fit our model to the given data by directly maximizing the likelihood. This is done without resorting to any approximations typically required in other nonlinear extensions of `LGM` such as the Particle Filter and the Extended/Unscented Kalman Filter.

2.2 Parameter Estimation using RNNs

In Sec. 2.1, we assumed the data was generated from a given family of `NKF` state space models characterized by its parameters $\Theta_{1:T} = (\mu_1, \Sigma_1, \{\Gamma_t, A_t\}_{t=1}^{T}, \{\Sigma_t, F_t\}_{t=2}^{T})$ along with Λ, the parameters of the normalizing flow transformation f_t (which we assume to be constant over time). However the exact values of the parameters are unknown. Similar to [9, 8], we propose to predict the parameters Θ from the covariates, using a recurrent neural network (RNN) where the recurrent function Ψ is parametrized by Φ, taking into account the possibly nonlinear relationship between covariates x_t:

$$\Theta_t = \sigma(h_t; \Phi), \quad h_t = \Psi(x_t, h_{t-1}; \Phi), \quad t = 1, \ldots T, \tag{5}$$

where σ denotes the transformation mapping of the RNN output to domains of the parameters (Appendix B.1). The RNN allows our model to be more expressive by allowing time-varying parameters along with temporal dependencies in the covariates, a requirement for forecasting. The parameters of the RNN Φ and of the normalizing flow Λ are estimated by maximizing the conditional likelihood,

$$p(y_{1:T}|x_{1:T}; \Phi, \Lambda) := p(y_{1:T}; \Theta_{1:T}, \Lambda) \tag{6}$$

where $p(y_{1:T}; \Theta_{1:T}, \Lambda)$ is given by (4). Although the transition model in our case is linear, complex dynamics can still be taken into account as the parameters of the `LGM`, which are now the outputs of a RNN, may change in a nonlinear way.

2.3 Applications: Forecasting and Missing Values

Given a model for sequential data, we can apply it to obtain future time steps $T{+}1{:}T{+}\tau$, or in other words, *forecasts* of the time series, starting from past observations $y_{1:T}$ and covariates $x_{1:T+\tau}$.

In our case such a forecast distribution $p(y_{T+1:T+\tau}|y_{1:T}, x_{1:T+\tau}; \Phi, \Lambda)$ can also be computed efficiently and is available in closed-form. From this, not only can we readily evaluate possible future scenarios (see Appendix A.4), but also draw samples from it to generate forecasts.

This is achieved by first computing the filtered distribution for the input range $1 : T$, and then recursively apply the transition equation and the observation model to generate prediction samples. More precisely, starting with the RNN state h_T and l_T sampled from $p(l_T|y_{1:T}, x_{1:T}; \Theta_{1:T})$, given by (5) and (6), respectively, we iteratively apply:

$$F_t, A_t, \Sigma_t, \Gamma_t = \sigma(h_t; \Phi), \qquad h_t = \Psi(x_t, h_{t-1}; \Phi), \tag{7a}$$

$$l_t = F_t l_{t-1} + \epsilon_t, \qquad \epsilon_t \text{ sampled from } \mathcal{N}(0, \Sigma_t), \tag{7b}$$

$$y_t = f_t(A_t^T l_t + \varepsilon_t), \qquad \varepsilon_t \text{ sampled from } \mathcal{N}(0, \Gamma_t), \qquad t = T+1, \ldots, T+\tau. \tag{7c}$$

In contrast to alternative deep learning approaches [1, 8, 11], this generative procedure is *not* autoregressive in the sense that observations y_t are never fed to the model. Instead, it is the filtering that correctly updates our beliefs based on the observed data. This has several notable consequences: we do not have to resort to large and cumbersome beam searches to obtain proper uncertainty estimates, noisy observations with varying levels of uncertainty can be properly handled, and long-term forecast computations are accelerated as intermediate observations need not be computed.

In many real world applications, observations for each time step may not be available, e.g., out-of-stock situations in the context of demand time series (no sales does not mean no demand if out-of-stock) or network failures in the case of sensor data streams. *Handling missing data* is then of central importance and there are two aspects to it: (i) learning in the presence of missing values without imputing them and (ii) imputing the missing values. Similar to LGM [20], our approach offers a straightforward, tractable and unbiased way for handling missing data in both of these scenarios.

In case of learning with missing values the likelihood terms corresponding to the missing entries should be ignored. This amounts to effectively dealing with them in the filtering step. Without loss of generality, let us assume that targets until time $t-1$ are observed but are missing for t. In this case, we can compute the filtered distribution $p(l_{t-1}|y_{1:t-1}; \Theta)$ with the method outlined in Sec. 2. As y_t is not observed, the filtered distribution at time t then simply corresponds to $p(l_t|y_{1:t-1}; \Theta)$, and can be obtained by applying the prediction step, starting from the filtered distribution at time $t-1$.

For the second case, we wish to impute missing values at time $t \notin \mathtt{t}^{\text{obs}}$ based on observed values at times $\mathtt{t}^{\text{obs}}$. This can be easily achieved by computing the smoothed distribution $p(l_t|y_{\mathtt{t}^{\text{obs}}}; \Theta)$ in the presence of missing data, and from this compute $p(y_t|y_{\mathtt{t}^{\text{obs}}}; \Theta)$ (full details in Appendix A.5). This distribution is available in closed form and can be readily sampled from. This allows us to use the same model for forecasting and imputation without the need for bidirectional RNNs [21] as the smoothing procedure inherently handles future data.

3 Local-Global Instantiation

The number of parameters of the NKF scales quadratically in the dimensionality d of the state (stemming from the covariance matrices of the Gaussian distributions) in the worst case. Leveraging the strength of NF, we present an instantiation of NKF with an induced *local-global* structure, that displays several advantages over the general unstructured form, e.g., exhibiting linear scaling in d.

We assume that each time series is associated with latent factors that evolve *independently w.r.t.* the factors of the other time series. These factors will in turn be *mixed* together with the normalizing flow, producing *dependent* time series observations. Formally, for each univariate time series i we associate a *local* LGM with parameters $\Theta^{(i)} = (\mu_1^{(i)}, \Sigma_1^{(i)}, \{\Gamma_t^{(i)}, A_t^{(i)}\}_{t\geq 1}, \{\Sigma_t^{(i)}, F_t^{(i)}\}_{t\geq 2})$, whose dynamics are characterized by:

$$\mathbf{l}_t^{(i)} = F_t^{(i)} \mathbf{l}_{t-1}^{(i)} + \boldsymbol{\epsilon}_t^{(i)}, \quad \boldsymbol{\epsilon}_t^{(i)} \sim \mathcal{N}(0, \Sigma_t^{(i)}), \tag{8a}$$

$$\mathbf{z}_t^{(i)} = A_t^{(i)T} \mathbf{l}_t^{(i)} + \boldsymbol{\varepsilon}_t^{(i)}, \quad \boldsymbol{\varepsilon}_t^{(i)} \sim \mathcal{N}(0, \Gamma_t^{(i)}), \tag{8b}$$

$$\mathbf{y}_t = f_t(\mathbf{z}_t^{(1)}, \ldots, \mathbf{z}_t^{(N)}). \tag{8c}$$

Note that here $\Gamma_t^{(i)}$ is a scalar and denotes variance of the Gaussian noise. This local-global structure has several advantages: (i) the dynamics of the local states need not be the same for each time series and thus, prior knowledge on the evolution can be readily injected *per* time series, (ii) computations can be done in parallel for each time series and finally (iii) we may benefit from the effects of amortization by predicting local parameters for each time series and sharing the same weights Φ:

$\Theta_t^{(i)} = \Psi(x_t^{(i)}, h_{t-1}^{(i)}; \Phi)$, allowing the RNN to make analogies between time series. In this case, dependencies across time series are captured with the normalizing flow f_t, mixing the components together in a nonlinear fashion.

We now explain a possible instantation of the `LGM` (Eq. (8a), (8b)), which is an innovation-based model, similar to [9]. Note that this is a choice and not a requirement as any other `LGM` instance may be used if that better reflects the data at hand. Nonetheless, with this form a wide range of phenomena can be captured, e.g., long term direction (trends), patterns that repeat (seasonality), cycles with different periodicity, multiplicative errors, etc.; we refer the reader to [5]. In our instantiation, we combine both level-trend and seasonality components, as described in Appendix B.2.

For f_t, we use the RealNVP architecture [13]: it is a network that is composed of a series of parametrized invertible transformations with a lower triangular Jacobian structure and vector component permutations in order to capture complex dependencies. This structure has the advantage of being flexible, while maintaining a tractable Jacobian term, computable in linear time. Moreover, as opposed to more recent architectures e.g., [14], the number of parameters scales linearly. For the RNN, we use an LSTM with 2 layers.

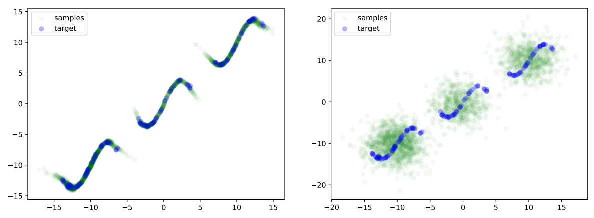
In terms of parameter complexity, this particular instantiation scales as $O(N)$ for each timestep, due to the diagonal structure of the covariance matrices (although they do not correspond to the effective number of parameters as these are predicted from the RNN). Moreover, time complexity for likelihood computation and forecasting scales as $O(N(d^2 + k))$ for each timestep, where k is the number of covariates and d is the dimension of latent state; in experiments $d = 32$ (24 hourly components + 7 daily components + one level component) for hourly data and $d = 8$ for daily data.

Partial observations: The local-global instantiation model presented above could deal with partial observations (i.e., only some entries of $y_t \in \mathbb{R}^N$ are missing but not all); however it would require marginalisation over the missing dimensions, which cannot be done in closed-form. Alternatively, if one is interested in dealing with partial observations, it is possible to consider an instantiation with a global (i.e., multivariate) `LGM` with non-diagonal covariance matrix modelling *linear* dependencies among the time-series $y_t \in \mathbb{R}^N$ at any given time step t, and a normalising flow applied locally to each time-series: $y_t = f_t(z_t) = (u_t(z_{1,t}), \dots, u_t(z_{N,t}))$, with $u_t : \mathbb{R} \to \mathbb{R}$ (see ablation study in our experiments). In this case, the marginalisation of any missing set of time series actually yields an analytic form for the filtering, smoothing and forecast distribution [20], and hence the handling of partial observations can be efficiently dealt with.

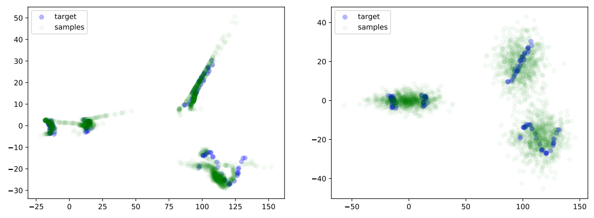
4 Experiments

4.1 Qualitative Results

We qualitatively assess our approach in Section 3 with two synthetic datasets `S1` and `S2` with increasing difficultly. The datasets are composed of 2 daily univariate time series with a weekly seasonality pattern. We compare our approach against a variant without any normalizing flow (this amounts to setting f_t = `id` in (8)). For dataset `S1` the daily data has three different modes and highly non-Gaussian time-*independent* observational noise. `S2` is similar, but the time series are *mixed* together in a nonlinear fashion, and the observational noise is not only non-Gaussian, but time-*dependent* (Appendix C.1 contains a detailed description). Target data (blue) and forecasts (green) are plotted in Fig. 2, where the first and second axis correspond to the first and second time series respectively, and data is aggregated over the temporal axis.



(a) `S1` : Results with (left) and without (right) NF.



(b) `S2` : Results with (left) and without (right) NF.

Figure 2: Evaluation w/ and w/o NF. The axes correspond to the two components of the 2D time series; the temporal axis is marginalized out. Green points correspond to model samples when predicting 24 days ahead and blue points to future samples from the data-generating process.

Approach	`VES`	`VAR`	`GARCH`	`DeepAR`	`GP-Copula`	`KVAE`	`NKF`(Ours)
Multivariate	✓	✓	✓	×	✓	✓	✓
Non-Linear, Non-Gaussian	×	×	×	✓	✓	✓	✓
Filtering & Smoothing	✓	NA	NA	NA	NA	✓	✓
Tractable Multi-step Forecast	✓	×	×	×	×	×	✓
Tractable Data Imputation	✓	×	×	×	×	×	✓

Table 1: Comparative summary of competing approaches on various parameters.

Observations and forecasts can also be seen in Appendix C.1 from a viewpoint that better highlights the seasonal nature of the observations.

For both datasets, the variant with $f_t = \texttt{id}$ captures the daily modes correctly, but assumptions such as Gaussianity and independence between time series introduce errors. In contrast, visual inspection reveals that applying the normalizing flow allows us to fit the data better, capturing the complex dependencies between time series and the non-Gaussian noise, for both `S1` and `S2`.

4.2 Quantitative Results

We follow the experimental set up proposed in [1] since it focusses on the same problem in the forecasting application. In particular, we evaluate on the public datasets used in [1]; see C.2.1 for details and Table 3 for the summary of datasets. Similar to [1], the forecasts of different methods are evaluated by splitting each dataset in the following fashion: all data prior to a fixed *forecast start date* compose the training set and the remainder is used as the test set. We measure the accuracy of the forecasts of various methods on all the time points of the test set. The hyperparameters have been selected using a validation set of equal size to the test set, created from the last time steps of the training data.

Our evaluation is extensive, covering relevant classical multivariate approaches as well as recent deep learning based models. In particular, we compare against `VES`, a direct generalization of univariate innovation state space model (a special case of `LGM` used in forecasting) to multivariate time series (see Chapter 17 of [5]), `VAR`, a multivariate linear autoregressive model and `GARCH` a multivariate conditional heteroskedastic model [22]; we include result of Lasso-regularized `VAR` as well. We also compare against the recent deep-learning based approaches `GP-Copula` [1] and `KVAE` [12] specifically developed for handling non-Gaussian multivariate data with non-linear dependencies. `GP-Copula` builds on ideas of `VAR` and relies on RNN and low-rank Gaussian Copula process for going beyond Gaussianity and linearity. In contrast, `KVAE` uses a variational autoencoder on top of linear state space models to achieve the same. Unlike our `NKF` model, inference and likelihood computation are not tractable in `KVAE` and it relies on particle filters for their approximation. Additionally, we compare with `DeepAR` [8] an autoregressive recurrent neural network based method for univariate time series forecasting. `DeepState` [9] is a special case of `NKF` model and is part of the ablation study. See Table 1 for summary of the compared methods based on various parameters.

In order to evaluate forecasting models, *continuous ranked probability score* (CRPS) is generally accepted as one of the most well-founded metrics[23, 24]. However, this metric is only defined for univariate timeseries and cannot assess if dependencies across time series are accurately captured. Different generalizations to the multivariate case have been used, e.g., the energy score, or CRPS-Sum [1]. We have opted for the CRPS-Sum, as the energy score suffers from the curse of dimensionality, from both a statistical and computational viewpoint [25, 26]. The introduction of the CRPS-Sum [1] was experimentally justified. In this work, we prove it is theoretically sound as justified formally in Appendix C.4 following the proper scoring rule framework [24]. Note that, opposed to [1], as different time series observations may have drastically different scales, we first normalize each time series by the sum of its absolute values before computing this metric (hence the '-N' suffix). We also did not choose log-likelihood since not all methods yield analytical forecast distributions and is not meaningful for some methods [1].

We report CRPS-Sum-N metric values achieved by all methods in Table 2. Classical methods, because of the Gaussianity and linear dependency assumptions, typically yield inferior results; entries marked with '-' are runs failed with numerical issues. Deep learning based models have superior

	method	exchange	solar	elec	wiki	traffic
	VES	**0.005 ± 0.000**	0.9 ± 0.003	0.88 ± 0.0035	-	0.35 ± 0.0023
	VAR	**0.005 ± 0.000**	0.83 ± 0.006	0.039 ± 0.0005	-	0.29 ± 0.005
	VAR-Lasso	0.012 ± 0.0002	0.51 ± 0.006	0.025 ± 0.0002	3.1 ± 0.004	0.15 ± 0.002
	GARCH	0.023 ± 0.000	0.88 ± 0.002	0.19 ± 0.001	-	0.37 ± 0.0016
	DeepAR	0.006±0.001	0.336±0.014	0.023±0.001	0.127±0.042	0.055±0.003
	GP-Copula	0.007±0.000	0.363±0.002	0.024±0.000	0.092±0.012	**0.051±0.000**
	KVAE	0.014 ± 0.002	0.34 ± 0.025	0.051 ± 0.019	0.095 ± 0.012	0.1 ± 0.005
	NKF(Ours)	**0.005 ± 0.000**	**0.320±0.020**	**0.016±0.001**	**0.071±0.002**	0.10±0.002
ablation study	$f_t = \texttt{id}$	0.005±0.000	0.415±0.002	0.026±0.000	0.082±0.000	0.123±0.000
	f_t Local	0.005±0.000	0.405±0.005	0.018±0.001	0.068±0.004	0.102±0.013

Table 2: CRPS-Sum-N (lower is better), averaged over 3 runs. The case $f_t = \texttt{id}$ is DeepState [9] and VES can be seen as part of ablation where normalizing flow and RNN are removed from NKF.

performance overall. In particular, NKF achieves the best result in 4 out of 5 datasets. On traffic NKF is better than all methods except for DeepAR and GP-Copula which are purely data-driven autoregressive approaches with minimal modelling assumptions. Given the domain of traffic dataset is $(0, 1)$, it would be interesting to verify in future work if the relatively weak performance is due to a short-coming of the normalizing flow part or due to the modelling choice of adopting a state space model instead of an autoregressive process.

Ablation Study First note that VES, which models only Gaussian data with linear dependencies, can be seen as an ablation where RNN and normalizing flow are not used. Next, to evaluate the usefulness of the normalizing flow in accurately modelling real world data, we analyze the performance of two particular instances of NKF : (i) '$f_t = \texttt{id}$': the normalizing flow is set to be the identity function, as in Section 4.1, therefore also reducing to the DeepState model proposed in [9] (ii) 'f_t Local': the normalizing flow is applied *locally* for each time series, modelling non-Gaussianity and non-linearity, but not any dependencies between time series, as explained in Appendix C.2.2. This ablation is important in order to analyze the advantages of capturing the potential dependencies across time series in the data. Overall, we observe (bottom lines in Table 2) a significant increase in performance from the identity function to a local NF, along with another increase when applying the global NF (see NKF results), apart from the wiki dataset. While we do not have a satisfying explanation, we speculate that this is due to modelling errors, the optimization algorithm, or simply because the time series may not exhibit much dependencies between each other.

Missing Data Experiment We evaluate our model in the context of varying degrees of missing data during training and evaluation. From elec, we remove $p = 10 + 20k, k = 0 \ldots, 4$, percent of the training data at random. This dataset, while realistic, is highly regular with clear seasonality patterns. The models are then evaluated on rolling windows, where the input range observations are missing with the same probability. In Fig. 3 and Appendix C.3, we report results for our approach, along with DeepAR, GP-Copula, KVAE. We observe that not only does NKF outperform other approaches by a large margin, but its error also increases slower than in other methods when the percentage of missing data is increased. We believe that this is due to the proper handling of the uncertainties in our approach since our model does not directly take observations as input. Moreover, the strong results obtained for up to 90% of missing data demonstrate that our method encodes useful prior knowledge due to the structure induced in the LGM, rendering this method useful even in low data regimes (the same observation is made in [9] for this dataset).

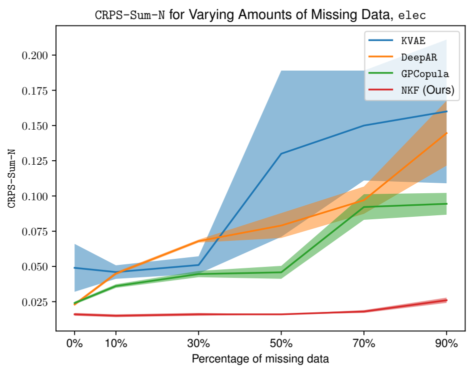


Figure 3: Forecasting with missing data.

5 Related Work

Neural networks for forecasting have seen growing attention in recent years [8, 27, 28, 29, 30, 31, 32]. We refer to [33] for an introductory overview. Most work concerns the univariate case using global models, assuming that time series are independent given the covariates and the model parameters. The family of global/local models, e.g., [34, 28], provide a more explicit way of incorporating global effects into univariate time series, without attempting to estimate the covariance structure of the data. An explicit probabilistic multivariate forecasting model is proposed in [1], which relies on Gaussian copulas to model non-Gaussian multivariate data.

Further related work combines probabilistic time series models with neural networks (e.g., point/renewal processes and neural networks [35] or exponential smoothing based expressions [27]). We extend the approach in [9] which uses an RNN to parametrize a state space model to the multivariate case alleviating Gaussianity and linear dependency assumptions in the observation model.

The idea to take advantage of the appealing properties of Kalman Filters (KF) [36] while relaxing its assumptions is not new. Prominent examples include the Extended Kalman Filter (EKF) [17], the Unscented Kalman Filter (UKF) [18] and Particle Filters (PF) [37] that relax the linearity and Gaussianity assumption by approximation or sampling techniques. The Gaussian process state space model (GPSSM) [38, 39] is a nonlinear dynamical system that extends `LGM`s by using GPs as the transition and/or observation mappings but typically assume the noise is additive Gaussian. If the noise is non-Gaussian, then these models again have to resort to approximation techniques similar to Particle Filters. Kernel Kalman Filters [40] address the linearity limitation of `LGM`s by defining the state space model in the reproducing Kernel Hilbert space (RKHS). In particular, the random latent state and the observation variable are mapped to RKHS and the state dynamics and the observation model are assumed to be linear in the kernel space. Note, however, that this approach still relies on the assumption that the noise is additive Gaussian.

Similarly, combining KF with neural networks is not new. Additionally to [9], [11] proposes to combine KF with Variational Auto-Encoders (KVAE) and [12] proposes variational approximations of the predictive distribution in nonlinear state space models. Finally, while most work on normalizing flows [13, 14, 41, 15] was presented in the i.i.d. setting, extensions to sequential data have recently been proposed [42, 43, 44]. Concurrent independent work by [45] also addresses the multivariate time series forecasting problem by combining normalizing flows with (deep) autoregressive models instead of state space models as in our work.

6 Discussion and Conclusion

In this paper we presented a simple, tractable and scalable approach to high-dimensional multivariate time series analysis, combining classical state space models with normalizing flows. Our approach can capture non-linear dependencies in the data and non-Gaussian noise, while still inheriting important analytic properties of the linear Gaussian state space model. This model is flexible, while still retaining interesting prior structural information, paramount to good generalization in low data regimes. Experimentally, our approach achieves the best results among a wide panel of competing methods on the tasks of forecasting and missing value handling. One caveat of our approach is that we no longer have identifiability w.r.t. the state space parameters: an interesting avenue of research is to work towards identifiability, e.g. by constraining the normalizing flow's expressivity.

Broader Impact

The present article stems from the authors' work on time series forecasting and anomaly detection in industrial settings. The methods proposed here are not tied to specific time series applications, but will likely be beneficial in supply chain and monitoring settings where large panels of time series data are commonly produced, data generation processes are too complex to be modelled fully and full automation is an aspirational goal.

Beneficiaries of applications of this work are therefore primarily companies with data gathering infrastructure and historical data. Such companies will be able to make their processes more efficient given better time series analytics as proposed here. Societal consequences will be similar to other cases where resources can be put to a more efficient usage: less waste, lower energy consumption, less need for human intervention. While this sounds generally appealing, e.g., from an environmental perspective, there is a price for increased efficiency which we are observing in the present time: lack of robustness in the face of disaster.

The current COVID-19 crisis has revealed how overly lean supply chains (e.g., for medical supplies) can result in shortages. This phenomenon is not new and has been observed in the automotive industry for example in 2011 after an earthquake and tsunami stroke in Japan [46]. We speculate that such phenomena may occur in other application scenarios as well where the reduction of buffers is enabled through better predictive accuracies as presented here at the consequence of worse disaster recovery (e.g., more efficient usage of cloud compute resources). A large discussion in society is needed how we should balance efficiency and robustness in critical areas and the identification of these critical areas.

The central assumption in our methodology is that the past is a meaningful indication of the future. This assumption is, when disaster occurs, violated. Hence, systems relying on methods as ours need to handle such violations gracefully (see [47] for an early example). At present, human intervention and overrides must be enabled in systems incorporating our method. This central assumption also means that potential biases in the data will be reproduced unless otherwise intervened.

Finally, we remark that multivariate time series models may be attractive to model epidemics as the present and that it may be tempting to try out our method on the high-dimensional data currently observed. We strongly advise against drawing conclusions from such experiments. The spreading of diseases is a well-understood process and interventions such as lock-downs need to be properly modelled and accounted for. Much further work is needed to allow such fine-grained analysis with our method and a naive application of the present method will almost surely result in unwanted results and unnecessary confusion.

References

[1] David Salinas, Michael Bohlke-Schneider, Laurent Callot, Roberto Medico, and Jan Gasthaus. High-dimensional multivariate forecasting with low-rank gaussian copula processes. In *Advances in Neural Information Processing Systems 32*, pages 6824–6834. 2019.

[2] Helmut Lütkepohl. *New introduction to multiple time series analysis*. Springer Science & Business Media, 2005.

[3] Ashton de Silva, Rob J Hyndman, and Ralph Snyder. The vector innovations structural time series framework: a simple approach to multivariate forecasting. *Statistical Modelling*, 10(4):353–374, 2010.

[4] Luc Bauwens, Sébastien Laurent, and Jeroen VK Rombouts. Multivariate garch models: a survey. *Journal of applied econometrics*, 21(1):79–109, 2006.

[5] Rob Hyndman, Anne Koehler, Keith Ord, and Ralph Snyder. *Forecasting with exponential smoothing. The state space approach*. 2008.

[6] James Durbin and Siem Jan Koopman. *Time series analysis by state space methods*, volume 38. Oxford University Press, 2012.

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

[8] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 2019.

[9] Syama Sundar Rangapuram, Matthias W Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. Deep state space models for time series forecasting. In *Advances in Neural Information Processing Systems*, pages 7785–7794, 2018.

[10] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015*, pages 1530–1538, 2015.

[11] Marco Fraccaro, Simon Kamronn, Ulrich Paquet, and Ole Winther. A disentangled recognition and nonlinear dynamics model for unsupervised learning. In *Advances in Neural Information Processing Systems 30*, pages 3601–3610. Curran Associates, Inc., 2017.

[12] Rahul G Krishnan, Uri Shalit, and David Sontag. Structured inference networks for nonlinear state space models. In *31st AAAI Conference on Artificial Intelligence*, 2017.

[13] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *5th International Conference on Learning Representations, ICLR 2017*, 2017.

[14] Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018*, pages 10236–10245, 2018.

[15] Junier Oliva, Avinava Dubey, Manzil Zaheer, Barnabas Poczos, Ruslan Salakhutdinov, Eric Xing, and Jeff Schneider. Transformation autoregressive networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3898–3907. PMLR, 10–15 Jul 2018.

[16] 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.

[17] Paul Zarchan and Howard Musoff. *Fundamentals of Kalman Filtering: A Practical Approach, Fourth Edition*. American Institute of Aeronautics and Astronautics, Inc., Reston, VA, 2015.

[18] Simon J Julier and Jeffrey K Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004.

[19] D. Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2011.

[20] R. H. Shumway and D. S. Stoffer. An approach to time series smoothing and forecasting using the EM algorithm. *Journal of Time Series Analysis*, 3(4):253–264, 1982.

[21] Wei Cao, Dong Wang, Jian Li, Hao Zhou, Lei Li, and Yitan Li. BRITS: bidirectional recurrent imputation for time series. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pages 6776–6786, 2018.

[22] Roy Van der Weide. Go-garch: a multivariate generalized orthogonal garch model. *Journal of Applied Econometrics*, 17(5):549–564, 2002.

[23] James E Matheson and Robert L Winkler. Scoring rules for continuous probability distributions. *Management science*, 22(10):1087–1096, 1976.

[24] Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007.

[25] Pierre Pinson and Julija Tastu. *Discrimination ability of the Energy score*. Number 15 in DTU Compute-Technical Report-2013. Technical University of Denmark, 2013.

[26] Aaditya Ramdas, Sashank Jakkam Reddi, Barnabás Póczos, Aarti Singh, and Larry A. Wasserman. On the decreasing power of kernel and distance based nonparametric hypothesis tests in high dimensions. In Blai Bonet and Sven Koenig, editors, *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, pages 3571–3577. AAAI Press, 2015.

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

[28] Yuyang Wang, Alex Smola, Danielle Maddix, Jan Gasthaus, Dean Foster, and Tim Januschowski. Deep factors for forecasting. In *International Conference on Machine Learning*, pages 6607–6617, 2019.

[29] Jan Gasthaus, Konstantinos Benidis, Yuyang Wang, Syama S. Rangapuram, David Salinas, Valentin Flunkert, and Tim Januschowski. Probabilistic forecasting with spline quantile function RNNs. *AISTATS*, 2019.

[30] Nikolay Laptev, Jason Yosinsk, Li Li Erran, and Slawek Smyl. Time-series Extreme Event Forecasting with Neural Networks at Uber. In *ICML Time Series Workshop*. 2017.

[31] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long- and short-term temporal patterns with deep neural networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 95–104. Association for Computing Machinery, 2018.

[32] Boris N Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting. *arXiv preprint arXiv:1905.10437*, 2019.

[33] Konstantinos Benidis, Syama Sundar Rangapuram, Valentin Flunkert, Bernie Wang, Danielle Maddix, Caner Turkmen, Jan Gasthaus, Michael Bohlke-Schneider, David Salinas, Lorenzo Stella, et al. Neural forecasting: Introduction and literature overview. *arXiv preprint arXiv:2004.10240*, 2020.

[34] Rajat Sen, Hsiang-Fu Yu, and Inderjit S Dhillon. Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting. In *Advances in Neural Information Processing Systems 32*, pages 4838–4847. Curran Associates, Inc., 2019.

[35] Ali Caner Turkmen, Yuyang Wang, and Tim Januschowski. Intermittent demand forecasting with deep renewal processes. *arXiv preprint arXiv:1911.10416*, 2019.

[36] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[37] Jun S Liu and Rong Chen. Sequential monte carlo methods for dynamic systems. *Journal of the American statistical association*, 93(443):1032–1044, 1998.

[38] J. Ko and D. Fox. Learning GP-Bayes filters via Gaussian process latent variable models. *Autonomous Robots*, 30:3–23, 2011.

[39] M. P. Deisenroth, R. D. Turner, M. F. Huber, U. D. Hanebeck, and C. E. Rasmussen. Robust filtering and smoothing with gaussian processes. *IEEE Transactions on Automatic Control*, 57(7):1865–1871, 2012.

[40] Gregor H. W. Gebhardt, Andras Kupcsik, and Gerhard Neumann. The kernel Kalman rule — Efficient nonparametric inference with recursive least squares. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, pages 3754–3760, 2017.

[41] Jens Behrmann, Will Grathwohl, Ricky T. Q. Chen, David Duvenaud, and Jörn-Henrik Jacobsen. Invertible residual networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, pages 573–582, 2019.

[42] Junxian He, Graham Neubig, and Taylor Berg-Kirkpatrick. Unsupervised learning of syntactic structure with invertible neural projections. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

[43] Xuezhe Ma, Chunting Zhou, Xian Li, Graham Neubig, and Eduard Hovy. Flowseq: Non-autoregressive conditional sequence generation with generative flow, 2019.

[44] Zachary Ziegler and Alexander Rush. Latent normalizing flows for discrete sequences. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7673–7682, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[45] Kashif Rasul, Abdul-Saboor Sheikh, Ingmar Schuster, Urs Bergmann, and Roland Vollgraf. Multi-variate probabilistic time series forecasting via conditioned normalizing flows. *arXiv preprint arXiv:2002.06103*, 2020.

[46] David Simchi-Levi and Edith Simchi-Levi. We need a stress test for critical supply chains. *Harvard Business Review*, 2020.

[47] Michael Bohlke-Schneider, Shubham Kapoor, and Tim Januschowski. Resilient neural forecasting systems. In *Proceedings of DEEM*. ACM, 2020.

[48] 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. *Journal of Machine Learning Research*, 21(116):1–6, 2020.

[49] 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.

[50] Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository. `http://archive.ics.uci.edu/ml`, 2017.

[51] Gábor J Székely. E-statistics: The energy of statistical samples. *Bowling Green State University, Department of Mathematics and Statistics Technical Report*, 3(05):1–18, 2003.

Appendices

A Proofs

For completeness, we restate the `NKF` model:

$$\begin{aligned}\mathbf{l}_t &= F_t \mathbf{l}_{t-1} + \boldsymbol{\epsilon}_t, \quad \boldsymbol{\epsilon}_t \sim \mathcal{N}(0, \Sigma_t), \\ \mathbf{y}_t &= f_t(A_t^T \mathbf{l}_t + \boldsymbol{\varepsilon}_t), \quad \boldsymbol{\varepsilon}_t \sim \mathcal{N}(0, \Gamma_t).\end{aligned} \tag{9}$$

A.1 Filtering

Proposition 1 (Filtering)**.** The *filtered* distributions of the `NKF` model are *Gaussian* and are given by the filtered distributions of the corresponding `LGM` with pseudo-observations $z_t := f_t^{-1}(y_t),\ t \geq 1$. That is, $p(l_t|y_{1:t}; \Theta, \Lambda) = p_{\texttt{LGM}}\,(l_t|z_{1:t}; \Theta)$ where $p_{\texttt{LGM}}$ refers to the distribution given by the `LGM`.

Proof. Note that for simplicity we omit conditioning on the parameters. Let us first recall the recursive Bayesian estimation, consisting of two distinct steps, predict and update of the latent state l_t given by

$$\begin{aligned}\text{predict:} \quad & p(l_t|y_{1:t-1}) = \int p(l_t|l_{t-1}) p(l_{t-1}|y_{1:t-1}) \mathrm{d}l_{t-1}, \\ \text{update:} \quad & p(l_t|y_{1:t}) = \frac{p(y_t|l_t) p(l_t|y_{1:t-1})}{\int p(y_t|l_t) p(l_t|y_{1:t-1}) \mathrm{d}l_t}.\end{aligned} \tag{10}$$

The filtered distribution $p(l_t|y_{1:t})$ is obtained recursively applying both these steps until time t, starting from a prior on the state $p(l_1)$. However, in its current form one could expect that computing the latter would require approximating integrals, as equation 9 is non-linear and non-Gaussian, which is prohibitive in the high-dimensional setting. But we show by induction that filtered distributions are Gaussian and in fact coincide with the filtered distributions of the underlying linear Gaussian state space model.

Let $z_t := A_t l_t + \varepsilon_t$ so that $z_t = f_t^{-1}(y_t), \forall t = 1, 2, \ldots, T$. Since y_t is observed and f_t is an invertible, deterministic function, we can view z_t as the pseudo-observation generated from the underlying linear Gaussian state space model (LGM),

$$\begin{aligned} \mathbf{l}_t &= F_t \mathbf{l}_{t-1} + \boldsymbol{\epsilon}_t, \quad \boldsymbol{\epsilon}_t \sim \mathcal{N}(0, \Sigma_t), \\ \mathbf{z}_t &= A_t^T \mathbf{l}_t + \boldsymbol{\varepsilon}_t, \quad \boldsymbol{\varepsilon}_t \sim \mathcal{N}(0, \Gamma_t), \end{aligned} \tag{11}$$

By making use of the change of variables formula, the likelihood term in the update step of (10) can be written as:

$$p(y_t|l_t) = p_{\mathbf{z}_t}(z_t|l_t) D f_t^{-1}(y_t), \tag{12}$$

where $Dg(x) := |\det[\mathrm{Jac}_x(g)]|$ is the absolute value of the determinant of the Jacobian of g evaluated at x. By Eq. (11), $p_{\mathbf{z}_t}(z_t|l_t)$ is the density of the Gaussian variable $\mathrm{z}_t = A_t^T l_t + \varepsilon_t$ conditioned on l_t, and from this we obtain $\forall t = 1, 2, \ldots T$,

$$p(y_t|l_t) = \mathcal{N}(z_t|\, A_t^T l_t, \Gamma_t) D f_t^{-1}(y_t). \tag{13}$$

We proceed with inductive proof, first showing that the filtered distribution for the base case $t = 1$, $p(l_1|y_1)$, is Gaussian and the same as that of the underlying LGM. For this, we start with the first prediction step $p(l_1) = \mathcal{N}(\mu_1, \Sigma_1)$, which is assumed to be Gaussian. From Eq. (12), assuming $D f_1^{-1}(y_1)$ is non-zero, we get:

$$p(l_1|y_1) = \frac{p(l_1)p(y_1|l_1)}{\int p(l_1)p(y_1|l_1)dl_1} = \frac{p(l_1)p_{\mathbf{z}_1}(z_1|l_1)Df_1^{-1}(y_1)}{\int p(l_1)p_{\mathbf{z}_1}(z_1|l_1)Df_1^{-1}(y_1)dl_1} = \frac{p(l_1)p_{\mathbf{z}_1}(z_1|l_1)}{\int p(l_1)p_{\mathbf{z}_1}(z_1|l_1)dl_1} = p_{\texttt{LGM}}(l_1|z_1).$$

This is exactly the first filtered distribution for LGM with observation z_1 and is in fact Gaussian since $p(z_1|l_1)$ is Gaussian by Eq. (13) and $p(l_1)$ is assumed to be Gaussian. Let μ_1^f, Σ_1^f denote the mean and covariance of this filtered distribution. For the inductive step assume that $p(l_{t-1}|y_{1:t-1})$ is Gaussian with mean μ_{t-1}^f and variance Σ_{t-1}^f, and is the same as $p_{\texttt{LGM}}(l_{t-1}|z_{1:t-1})$. We will prove that $p(l_t|y_{1:t}) = p_{\texttt{LGM}}(l_t|z_{1:t})$.

As both $p(l_{t-1}|y_{1:t-1})$ and $p(l_t|l_{t-1}) = \mathcal{N}(l_t|F_t l_{t-1}, \Sigma_t)$ are Gaussian, it follows that the distribution obtained from the prediction step must also be Gaussian, i.e., $p(l_t|y_{1:t-1}) = \mathcal{N}(l_t|\mu_t^p, \Sigma_t^p), \forall t = 2, \ldots, T$, with:

$$\begin{aligned} \mu_t^p &= F_t \mu_{t-1}^f, \\ \Sigma_t^p &= F_t \Sigma_{t-1}^f F_t^T + \Sigma_t. \end{aligned} \tag{14}$$

In fact, this coincides with $p_{\texttt{LGM}}(l_t|z_{1:t-1})$ given that $\mu_{t-1}^f, \Sigma_{t-1}^f$ are the same for both LGM and our model and both use the same transition for the latent state.

Similar to the base case, the filtered distribution for $t > 1$ is

$$\begin{aligned} p(l_t|y_{1:t}) &= \frac{p(y_t|l_t)p(l_t|y_{1:t-1})}{\int p(y_t|l_t)p(l_t|y_{1:t-1})\mathrm{d}l_t} \\ &= \frac{Df_t^{-1}(y_t)p_{\mathbf{z}_t}(z_t|l_t)p(l_t|y_{1:t-1})}{\int Df_t^{-1}(y_t)p_{\mathbf{z}_t}(f_t^{-1}(y_t)|l_t)p(l_t|y_{1:t-1})\mathrm{d}l_t} \\ &= \frac{p_{\mathbf{z}_t}(z_t|l_t)p_{\texttt{LGM}}(l_t|z_{1:t-1})}{\int p_{\mathbf{z}_t}(z_t|l_t)p_{\texttt{LGM}}(l_t|z_{1:t-1})\mathrm{d}l_t} \\ &= p_{\texttt{LGM}}(l_t|z_{1:t}), \end{aligned} \tag{15}$$

which is the same as the filtered distribution of the corresponding LGM. In fact, one can deduce this filtered distribution in closed-form:

$$\begin{aligned} p(l_t|y_{1:t}) &= \frac{\mathcal{N}(z_t|\, A_t^T l_t, \Gamma_t)\mathcal{N}(l_t|\mu_t^p, \Sigma_t^p)}{\int \mathcal{N}(z_t|\, A_t^T l_t, \Gamma_t)\mathcal{N}(l_t|\mu_t^p, \Sigma_t^p)\mathrm{d}l_t} \\ &= \mathcal{N}(l_t|\, \mu_t^f, \Sigma_t^f), \end{aligned} \tag{16}$$

where $\mu_t^f = \mu_t^p + K_t[f^{-1}(y_t) - A_t^T \mu_t^p]$, $\Sigma_t^f = (I - K_t A_t^T)\Sigma_t^p$, and $K_t = \Sigma_t^p A_t (A_t^T \Sigma_t^p A_t + \Gamma_t)^{-1}$. This recursive formula for the filtered distribution is valid for $t > 1$ and the same for the base case $t = 1$ is obtained by noting that $\mu_1^p = \mu_1$ and $\Sigma_1^p = \Sigma_1$. $\square$

A.2 Smoothing

Proposition 2 (Smoothing)**.** The *smoothed* distributions of the NKF model are *Gaussian* and are given by the smoothed distributions of the corresponding LGM with pseudo-observations $z_t := f_t^{-1}(y_t), t = 1, 2 \dots, T$. That is, $p(l_t|y_{1:T}; \Theta, \Lambda) = p_{\texttt{LGM}}\,(l_t|z_{1:T}; \Theta)$.

Proof. Note that for simplicity, we omit conditioning on the parameters. This can again be proved by induction starting with the base case $t = T$ and running backwards. For the base case the smoothed distribution $p(l_T|y_{1:T})$ coincides with the filtered distribution for the final time step T, which was already shown to be equal to that of the standard LGM. For the inductive step assume that in time step $t+1$ it holds that $p(l_{t+1}|y_{1:T}) = p_{\texttt{LGM}}\,(l_{t+1}|z_{1:T})$. We will prove that the same is true in time step t, i.e., $p(l_t|y_{1:T}) = p_{\texttt{LGM}}\,(l_t|z_{1:T})$.

$$\begin{aligned}
p(l_t|y_{1:T}) &= \int p(l_t, l_{t+1}|y_{1:T})\mathrm{d}l_{t+1} \\
&= \int p(l_{t+1}|y_{1:T})p(l_t|l_{t+1}, y_{1:T})\mathrm{d}l_{t+1} \\
&= \int p(l_{t+1}|y_{1:T})p(l_t|l_{t+1}, y_{1:t})\mathrm{d}l_{t+1} \\
&= p(l_t|y_{1:t}) \int \frac{p(l_{t+1}|y_{1:T})p(l_{t+1}|l_t)}{p(l_{t+1}|y_{1:t})}\mathrm{d}l_{t+1} \\
&= p_{\texttt{LGM}}(l_t|z_{1:t}) \int \frac{p_{\texttt{LGM}}(l_{t+1}|z_{1:T})p(l_{t+1}|l_t)}{p_{\texttt{LGM}}(l_{t+1}|z_{1:t})}\mathrm{d}l_{t+1} \\
&= p_{\texttt{LGM}}(l_t|z_{1:T}),
\end{aligned} \tag{17}$$

where in the penultimate step we used the fact that the predictive distribution $p(l_{t+1}|y_{1:t})$ and the filtered distribution $p(l_t|y_{1:t})$ of our model are the same as those of the standard LGM (see Proposition 1 and (14)). The smoothed distribution of the standard LGM is Gaussian with mean μ_t^s and variance Σ_t^s such that, starting from $\mu_T^s = \mu_T^f, \Sigma_T^s = \Sigma_T^f$:

$$\mu_t^s = \mu_t^f + G_t[\mu_{t+1}^s - \mu_{t+1}^p], \tag{18a}$$

$$\Sigma_t^s = \Sigma_t^f + G_t[\Sigma_{t+1}^s - \Sigma_{t+1}^p], \tag{18b}$$

$$G_t = \Sigma_t^f A_t [\Sigma_{t+1}^p]^{-1}, \tag{18c}$$

where μ_t^f, Σ_t^f correspond to mean and covariance computed during the update step of the NKF, and μ_t^p, Σ_t^p computed in the prediction step (refer to Eq. (14)).

□

A.3 Likelihood

Proposition 3 (Likelihood)**.** The likelihood of the parameters (Θ, Λ) of the NKF model given the observations $\{y_{1:T}\}$ can be computed as

$$\ell(\Theta, \Lambda) = p(y_{1:T}; \Theta, \Lambda) = \prod_{t=1}^{T} p_{\texttt{LGM}}(z_t|z_{1:t-1}; \Theta) \left|\det\left[\mathrm{Jac}_{z_t}(f_t)\right]\right|^{-1}, \tag{4}$$

where $z_t = f_t^{-1}(y_t)$ and $p_{\texttt{LGM}}(z_t|z_{1:t-1}; \Theta)$ denotes the predictive distribution of LGM.

Proof. We can compute the likelihood by decomposing it into telescoping conditional distributions and using the substitution $z_t = f^{-1}(y_t)$,

$$\begin{aligned}
p(y_{1:T};\Theta,\Lambda) &= \prod_{t=1}^{T} p(y_t|y_{1:t-1};\Theta,\Lambda) \\
&= \prod_{t=1}^{T} \int \left|\det\left[\mathrm{Jac}_{y_t}(f_t^{-1})\right]\right| p_{\mathbf{z}_t}(z_t|l_t)p(l_t|y_{1:t-1};\Theta)\mathrm{d}l_t \\
&= \prod_{t=1}^{T} \left|\det\left[\mathrm{Jac}_{y_t}(f_t^{-1})\right]\right| \int p_{\mathbf{z}_t}(z_t|l_t)p_{\texttt{LGM}}(l_t|z_{1:t-1};\Theta)\mathrm{d}l_t \qquad (19)\\
&= \prod_{t=1}^{T} \left|\det\left[\mathrm{Jac}_{y_t}(f_t^{-1})\right]\right| p_{\texttt{LGM}}(z_t|z_{1:t-1};\Theta) \\
&= \prod_{t=1}^{T} \left|\det\left[\mathrm{Jac}_{y_t}(f_t^{-1})\right]\right| \mathcal{N}(z_t;\nu_t^p,\Gamma_t^p),
\end{aligned}$$

where in the third step we used the fact that predictive distributions of our model (9) and the standard `LGM` are the same. This is true because the filtered distributions of our model and the `LGM` are the same since both use the same transition for the latent state, as shown in Proposition 1 (see (14)). In the final step we used the fact that the predictive distribution $p(z_t|z_{1:t-1})$ of the standard `LGM` is Gaussian with mean ν_t^p and covariance Γ_t^p given by the analytical expressions [19]:

$$\begin{aligned}
\nu_t^p &= A_t^T F_t \mu_{t-1}^f, & \Gamma_t^p &= A_t^T \left(F_t \Sigma_{t-1}^f F_t^T + \Sigma_t\right) A_t + \Gamma_t, & t &> 1,\\
\nu_1^p &= A_t^T \mu_1, & \Gamma_1^p &= A_t^T \Sigma_1 A_t + \Gamma_1, & t &= 1.
\end{aligned}$$

□

A.4 Forecasting Distribution

The joint forecasting distribution of the future time steps $T+1:T+\tau$ can be obtained in terms of the predictive distribution of the corresponding `LGM` with $z_t = f^{-1}(y_t)$,

$$p(y_{T+1:T+\tau}|y_{1:T},x_{1:T+\tau};\Theta,\Lambda) = \prod_{t=T+1}^{T+\tau} p_{\texttt{LGM}}(z_t|z_{1:t-1};\Theta)\left|\det\left[\mathrm{Jac}_{z_t}(f_t)\right]\right|^{-1}. \qquad (20)$$

The exact analytical expressions for the forecasting distribution is given by:

$$\begin{aligned}
p(y_{T+1},\ldots,y_{T+\tau}|y_1,\ldots,y_T,\Theta) &= \prod_{t=T+1}^{T+\tau} \mathcal{N}(f^{-1}(y_t);\nu_t^p,\Gamma_t^p)\left|\det\left[\mathrm{Jac}_{y_t}(f_t^{-1})\right]\right|, \qquad (21)\\
\nu_t^p = A_t^T \mu_t^p, &\qquad \mu_t^p = F_t \mu_{t-1}^f, \qquad (22)\\
\Gamma_t^p = A_t^T \Sigma_t^p A_t + \Gamma_t, &\qquad \Sigma_t^p = F_t \Sigma_{t-1}^f F_t^T + \Sigma_t, \qquad (23)
\end{aligned}$$

where (μ_t^p, Σ_t^p) and (ν_t^p, Γ_t^p) are the parameters of the predictive distributions for the latent state and observations given in Propositions 1 and 3, respectively.

A.5 Handling Missing Data

Given a subset of observed targets $y_{\mathfrak{t}^{\text{obs}}}$, what can we say about the missing observations? Said otherwise, what is the probability of $p(y|y_{\mathfrak{t}^{\text{obs}}})$? This data imputation problem is of central importance in numerous applications. This problem can be solved by first solving the smoothing problem in order to obtain the posterior $p(l_t|y_{\mathtt{t}^{\text{obs}}})$:

$$
\begin{aligned}
p(y|y_{\mathtt{t}^{\text{obs}}}) &= \int p(y,l|y_{\mathtt{t}^{\text{obs}}})\mathrm{d}l \\
&= \int p(y|l)p(l|y_{\mathtt{t}^{\text{obs}}})\mathrm{d}l \\
&= \prod_t p_{\mathtt{LGM}}(z_t|z_{\mathtt{t}^{\text{obs}}})\left|\det\left[\mathrm{Jac}_{z_t}(f_t)\right]\right|^{-1} \\
&= \prod_t \mathcal{N}(z_t|A_t^T\mu_t^{u,s}, A_t^T\Sigma_t^{u,s}A_t + \Gamma_t)\left|\det\left[\mathrm{Jac}_{z_t}(f_t)\right]\right|^{-1},
\end{aligned}
\tag{24}
$$

where $(\mu_t^{u,s}, \Sigma_t^{u,s})$ corresponds to the parameters of the smoothed distribution in the presence of missing data. Once again, this distribution admits an analytical expression and can be readily sampled from.

B Model: Additional Details

B.1 Encoding of the LGM Parameters

In order to constrain the real-valued outputs h_t at time t of the RNN Ψ to the parameter domains of the LGM, we apply a sequence of transformations. For the j-th state space model parameter, Θ_t^j, we initially compute the affine transformation $\tilde{\Theta}_t^j = w_j^\top h_t + b_j$, where the weights and biases are different for each parameter and are all included in Φ and learned. We then transform $\tilde{\Theta}_t^j$ to the domain of the parameter by applying:

- for real-valued parameters: no transformation i.e., $\Theta_t^j = \tilde{\Theta}_t^j$,
- for positive parameters: the softplus function $\Theta_t^j = \log(1+\exp(\tilde{\Theta}_t^j))$,
- for bounded parameters in $[a,b]$: a scaled and shifted sigmoid $\Theta_t^j = (b-a)\frac{1}{1+\exp(-\tilde{\Theta}_t^j)} + a$.

In practice, it is often advisable to impose stricter bounds than theoretically required, e.g., enforcing an upper bound on the observation noise variance or a lower bound on the innovation strengths can stabilize the training procedure in the presence of outliers.

B.2 Local-Global Instantiation

As in [9], the evolution of the latent state $\mathbf{l}_t^{(i)}$ for each time series i is captured using a *composition* of level-trend and seasonality model, described below.

Local Level-Trend Model In the level-trend model, the latent state has two dimensions and is characterized by:

$$
\begin{aligned}
F_t^{(i)} &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \quad A_t^{(i)} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \\
\Sigma_t^{(i)} &= \begin{bmatrix} \alpha_t^2 & 0 \\ 0 & \beta_t^2 \end{bmatrix}, \quad \Gamma_t^{(i)} \in \mathbb{R}_{++}.
\end{aligned}
\tag{25}
$$

Local Seasonality Model In the case of seasonality-based models, each seasonality pattern can be described by a set of seasonal factors (or seasons) associated with it. For example, in the day-of-week pattern there are seven factors, one for each day of the week. We can represent each factor as a component of the latent state $\mathbf{l}_t \in \mathbb{R}^7$. Then, for the day-of-week seasonality model, we have

$$
\begin{aligned}
F_t^{(i)} &= I, \quad A_t^{(i)} = \mathbb{1}_{\{\text{day(t)=j}\}_{j=1}^7}, \\
\Sigma_t^{(i)} &= \sigma_t^2 \text{diag}(A_t^{(i)}), \quad \Gamma_t^{(i)} \in \mathbb{R}_{++}.
\end{aligned}
\tag{26}
$$

Composite Model We concatenate the state of the level-trend model and the seasonality model in order to take into account both types of dynamics.

Note that in order to take into account the correlations across time series, the *pseudo-observations* generated by the composite state space model are given to the normalizing flow f_t, which will implicitly capture these correlations.

C Experimental Details

We use the same hyperparameters for the model architecture across all datasets. For the RNN, we use an LSTM with the same architecture and hyperparameters as those proposed in `DeepState` [9], based on the open-source implementation from [48]. To avoid numerical issues arising during the `NKF` update step, we find it useful to lower bound the observation noise $\Gamma_t^{(i)}$ to 0.1 for `elec`, `solar`, `wiki`, and 0.01 for the rest. The numerical issues are perhaps due to the overfitting of `LGM` parameters to the training data; the open-source implementation [48] of `DeepState` [9] also recommends such safe guards on the noise terms. Note that in the experiments f_t is the same across all time steps; however time-dependant noise may still be captured as the parameters of the `LGM` are time-dependent. An interesting research avenue for future work may be to consider a time-varying normalizing flow by conditioning it on temporal features, thus bringing us to consider conditional NFs [10] .

We evaluated various NF proposed in the literature: *RealNVP* [13], *Glow* [14] and *iResnet* [41]. We have finally opted for RealNVP which is straightforward to implement and does not suffer from drawbacks of the other methods. In particular, the number of parameters in Glow scales quadratically in the number of dimensions due to the fully connected layers in the permutation step: for the wiki dataset, as the dimension of the observations is 2×10^3, this would imply that just one permutation layer would have 4×10^6 parameters. In iResnet, the estimator of the Jacobian term yielded a high variance in the high dimensional setting and requires a number of forward passes in order to compute the inverse. For all the experiments, we have set the number of blocks of the RealNVP to 9. Each affine-coupling is parameterized by a 2-layer neural network with the numbers of hidden dimensions set to 16, without batch-normalization.

During training we tune the number of epochs for each dataset on a separate validation set of equal size to the test set. We use the Adam optimizer with 2×10^{-4} learning rate and 1×10^{-6} weight-decay.

As in [1], the evaluation is done in a rolling fashion: for hourly datasets accuracy is measured on 7 rolling time windows where each roll corresponds to 24 hours, thus covering 7 days of the test set. For all the other datasets we use 5 windows. More details on the forecast horizon τ, domain, frequency, dimension N and length of training timesteps T are given in the Appendix C.2.1.

C.1 Qualitative Experiments

C.1.1 Datasets

Both datasets are composed of 2 daily univariate time series with weekly seasonality, correlated in different ways. The time series observations $y_{1:T}$, with $T = 120$, are generated according to the following state space model:

$$\begin{aligned} \mathbf{l}_t &= \mathbf{l}_{t-1} + \boldsymbol{\epsilon}_t, \quad \boldsymbol{\epsilon}_t \sim \mathcal{N}(0, 10^{-3} \times I), \\ \mathbf{y}_t &= m(A_t^T \mathbf{l}_t) + \boldsymbol{\eta}_t, \quad \boldsymbol{\eta}_t \sim \mathcal{D}(t), \end{aligned} \tag{27}$$

where $\mathbf{l}_t = [\mathbf{l}_t^{(1)}; \mathbf{l}_t^{(2)}] \in \mathbb{R}^{14}$ and each component of $\mathbf{l}_t^{(i)} \in \mathbb{R}^7$ is associated to a day of the week, $A_t \in \mathbb{R}^{14\times 2}$ is block diagonal where $A_t^{(i)} = \mathbb{1}_{\{\text{day(t)=j}\}_{j=1}^{7}}$ *selects* the corresponding state component based on t, $m : \mathbb{R}^2 \to \mathbb{R}^2$ is a *mixing* function, and $\mathcal{D}(t)$ is a highly non-Gaussian, time-dependent distribution. The initial state $l_1^{(i)}$ is sampled from a Gaussian distribution $\mathcal{N}(\mu_0, 10^{-3} \times I)$, with 3 different mean levels according to the day of the week[5]: $\mu_1 = [-10, -10, 0, 0, 0, 10, 10]^T$.

For the first experiment (corresponding to Figure 2a) we consider the simple case where no mixing occurs, i.e., m is the identity function. The observational noise $\boldsymbol{\eta}_t$ is the same for every t and highly

[5]Note that in our notation (see Eq. (1)), the prior state $\mathbf{l}_1$ is indexed by 1 not 0.

non-Gaussian. To this respect, $\boldsymbol{\eta}_t^{(1)}$ follows an 1D Uniform distribution and $\boldsymbol{\eta}_t^{(2)}$ is the image of $\boldsymbol{\eta}_t^{(1)}$ when mapped through a cosine function.

For the second experiment (corresponding to Figure 2b) we consider a more complex setting where mixing occurs and the observational noise $\boldsymbol{\eta}_t$ is time-dependent. Artificial dependencies across time series are induced, setting the *mixing* function m to $m([x, y]^T) = [x\, y, x + y]^T$. The observational noise $\boldsymbol{\eta}_t$ follows a day-of-week pattern, where three non-Gaussian distributions are selected based on the day of the week, similarly as for μ_1: for the first two days $\boldsymbol{\eta}_t$ is the same in the first experiment, for the three next days $\boldsymbol{\eta}_t$ follows a mixture of two Gaussians, and for the weekend $\boldsymbol{\eta}_t$ is a simple line generated from an 1D Uniform distribution.

C.1.2 Forecast Samples

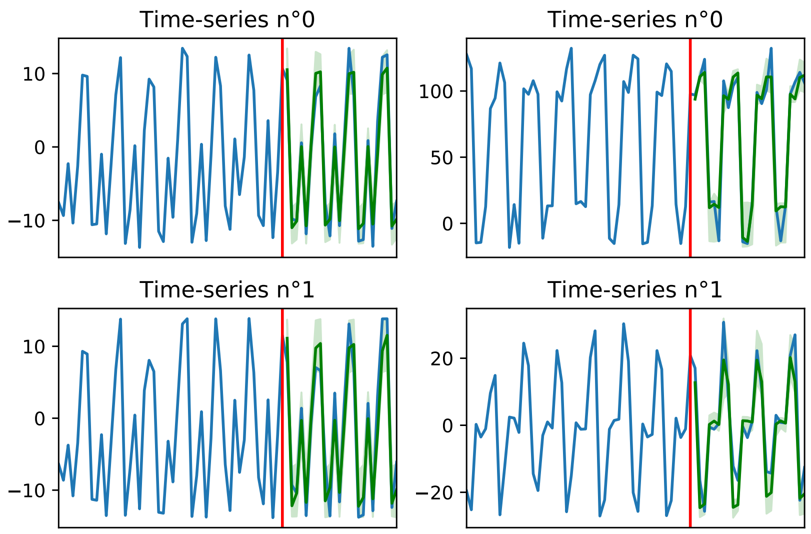


Figure 4: Forecast results with NF (`NKF` model) for both time-series of dataset `S1` (left) and `S2` (right). For each forecast: In blue, the values of the input (left of red line) and target (right of red line) time series. In dark green, the forecasts mean, and quantiles $[0.1, 0.9]$ are filled light green.

C.2 Quantitative Experiments

C.2.1 Datasets

We evaluate on the public datasets (with the same training and test splits) used in [1]: `exchange`: daily exchange rate between 8 currencies as used in [49]; `solar`: hourly photo-voltaic production of 137 stations in Alabama State used in [49]; `elec`: hourly time series of the electricity consumption of 370 customers [50]; `traffic`: hourly occupancy rate, between 0 and 1, of 963 San Francisco car lanes [50]; `wiki`: daily page views of 2000 Wikipedia pages used in [29]. Similar to [1], the forecasts of different methods are evaluated by splitting each dataset in the following fashion: all data prior to a fixed *forecast start date* compose the training set and the remainder is used as the test set. We measure the accuracy of the forecasts of various methods on all the time points of the test set.

In Table 3 we summarize the details of the datasets used to evaluate the models.

C.2.2 Ablation Study: f_t `Local`

Here we give additional details for the local variant of our model `NKF-Local`. This variant uses the same form for the state-space model (Eq. (8a) and (8b)), with an alternative local normalizing flow $u_t : \mathbb{R} \to \mathbb{R}$, applied to each time-series independently:

$$f_t(z_t) = (u_t(z_{1,t}), \ldots, u_t(z_{N,t})). \tag{28}$$

dataset	τ (num steps predicted)	domain	frequency	dimension N	time steps T
`exchange`	30	$\mathbb{R}^+$	daily	8	6071
`solar`	24	$\mathbb{R}^+$	hourly	137	7009
`elec`	24	$\mathbb{R}^+$	hourly	370	5790
`traffic`	24	$\mathbb{R}^+$	hourly	963	10413
`wiki`	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.

In this case, the conditional density in Eq. (2) of variables formula can easily be expressed in terms of normalizing flow u_t, and reduces to:

$$\begin{aligned} p(y_t|l_t;\Theta,\Lambda) &= p(y_{1,t},\ldots,y_{N,t}|l_t;\Theta,\Lambda) \\ &= p_{z_t}(u_t^{-1}(y_{1,t}),\ldots,u_t^{-1}(y_{N,t});\Theta)\prod_{i=1}^{N}\left|\det\left[\mathrm{Jac}_{y_{t,i}}(u_t^{-1})\right]\right|. \end{aligned} \tag{29}$$

Written in this form, we can see that the computation of the Jacobian term scales linearly in the dimension, as $y_{i,t} \in \mathbb{R}$ and can be calculated in parallel.

For the univariate u_t , we use non-time dependent iResnet [41], with 6 invertible blocks, LipSwish activation, spectral normalizing coefficient of 0.9, and 10 fixed-point iterations for the computation of the inverse, and 1 iteration for the power iteration method. We set the learning rate associated to the RNN to 0.001, and 0.00001 for the normalizing flow.

C.3 Handling Missing Data

Here we report the exact numbers for the missing data experiment, Table 4.

	`elec`10%	`elec`30%	`elec`50%	`elec`70%	`elec`90%
`KVAE`	0.046 ± 0.0048	0.051 ± 0.0062	0.13 ± 0.059	0.15 ± 0.039	0.16 ± 0.051
`DeepAR`	0.045±0.001	0.068±0.001	0.079±0.009	0.097±0.010	0.145±0.023
`GP-Copula`	0.036±0.0013	0.045±0.002	0.046±0.0046	0.092±0.0091	0.094±0.0077
`NKF`	**0.015±0.001**	**0.016±0.001**	**0.016±0.000**	**0.018±0.001**	**0.026±0.0019**

Table 4: CRPS-Sum-N (lower is better), averaged over 3 runs.

C.4 A Note on Evaluation Multivariate Metrics for Probabilistic Forecasting

CRPS relies on the pinball loss that measures the fit at each quantile between the quantile function F^{-1} and an observation:

$$\Lambda_\alpha(q,y) = (\alpha - \mathbb{1}_{\{y<q\}})(y-q), \tag{30}$$

where $\alpha \in [0,1]$ is the quantile level and q the respective quantile of the probability distribution.

The integrated pinball loss over all quantile levels $\alpha \in [0,1]$ is defined as the CRPS:

$$\text{CRPS}(F^{-1},y) = \int_0^1 2\Lambda_\alpha(F^{-1}(\alpha),y)\,d\alpha. \tag{31}$$

We estimate F^{-1} by drawing 100 samples and sorting them.

Although CRPS is a widely accepted metric for assessing the quality of probabilistic forecasts in the univariate case, it is not applicable in the multivariate setting as it does not capture correlations across time-series. Instead, [1] introduced CRPS-Sum an extension of CRPS to the multivariate case:

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

where F^{-1} is estimated by summing the samples across dimensions and then computing the quantiles by sorting. However, [1] does not give a theoretical justification of CRPS-Sum.

Below, we prove that CRPS-Sum is a proper scoring rule.

We restate fundamental definitions and results from [24] on proper scoring rules first before showing that CRPS-Sum-N is a proper scoring rule.

Let Ω be the sample space, let $\mathcal{A}$ be a σ-algebra on Ω, and let $\mathcal{P}$ be a convex class of probability measures on $(\Omega, \mathcal{A})$.

Definition 1. A *scoring rule* is a function $S : \mathcal{P} \times \Omega \to [-\infty, \infty]$, such that for every $P \in \mathcal{P}$ we have that $E(S(P, \cdot))$ exists (and is possibly non-finite). For such an S, and for every $P, Q \in \mathcal{P}$, define

$$S(P,Q) := \int S(P,\cdot)dQ.$$

Properness is defined thusly:

Definition 2. A scoring rule is proper with respect to $\mathcal{P}' \subset \mathcal{P}$ if $\forall P, Q \in \mathcal{P}'$ we have that $S(P,P) \geq S(Q,P)$. It is called strictly proper with respect to $\mathcal{P}'$ if equality holds iff $P = Q$ a.s.

Theorem 3. *Let $\Omega = \mathbb{R}$, and $\mathcal{A}$ be the Borel σ-algebra. Then the function*

$$\mathrm{CRPS}(P,x) := -\int_{\mathbb{R}} (P(\{t|t \leq y\}) - \mathbb{1}_{x \leq y})^2 dy$$

is a proper scoring rule with respect to the set $\mathcal{P}$ of probability measures on $\mathcal{A}$. Furthermore, if we restrict to the subclass $\mathcal{P}'$ of probability measures with finite first moment, then it can be shown that

$$\mathrm{CRPS}(P,x) = \frac{1}{2}E_P(|X - X'|) - E_P(|X - x|),$$

where X and X' are understood as independent random variables that have the distribution P, and CRPS *is strictly proper with respect to $\mathcal{P}'$.*

Note 4.

1. The term "$E_P(|X - X'|)$ where X and X' are understood as independent random variables that have the distribution P" means, by definition: $\int_{\Omega \times \Omega} |p_1 - p_2| d(P \times P)$, where p_i is the projection to the i^{th} coordinate.

2. It is crucial in the definition of properness that we require that $\forall P, Q \in \mathcal{P}'$ we have that $S(P,P) \geq S(Q,P)$ rather than $S(P,P) \geq S(P,Q)$. Indeed, it is easy to see that if P follows a standard normal distribution and Q is the constant distribution 0, then

$$\mathrm{CRPS}(P,Q) - \mathrm{CRPS}(P,P) =$$
$$E_P(|X - X'|) - E_P(|X|) =$$
$$\frac{2}{\sqrt{\pi}} - \sqrt{\frac{2}{\pi}} > 0.$$

3. The proof of strict properness in Theorem 3 with respect to $\mathcal{P}'$ boils down to the statement that for any independent X and X' following a probability distribution $P \in \mathcal{P}'$, and Y and Y' independent of each other and of X and X', following a probability distribution $Q \in \mathcal{P}'$, we have that:

$$2E(|X - Y|) - E(|X - X'|) - E(|Y - Y'|) =$$
$$\int (P(\{t|t \leq y\}) - Q(\{t|t \leq y\}))^2 dy,$$

and is therefore non-negative, and zero iff $P = Q$ a.s. An elementary proof can be found in pages 5 and 6 of [51].

In the case that $\Omega = \mathbb{R}^d$ and $\mathcal{A}$ is its associated Borel σ-algebra, we introduce the following new scoring rule.

Definition 5. For any choice of a measurable function $L : \mathbb{R}^d \to \mathbb{R}$ w.r.t to the Borel σ-algebras, define:

$$\mathrm{CRPS}(L, P, x) := \mathrm{CRPS}(L_*P, L(x)),$$

where L_*P is the pushforward measure of P by L.

It is trivial to verify that for every such choice of L, the function $\mathrm{CRPS}(L, \cdot, \cdot)$ is a scoring rule. Also note that the scoring rule $CRPS - Sum - N$ from Appendix G.1 of [1] is none other than $\mathrm{CRPS}(L, \cdot, \cdot)$ for L defined by

$$L(x_1, ..., x_d) = x_1 + ... + x_d.$$

In what follows we will not restrict to this choice of L.

Lemma 6. *The following equality holds for all probability measures P and Q:*

$$\mathrm{CRPS}(L, P, Q) = \mathrm{CRPS}(L_*P, L_*Q).$$

Proof. This boils down to the change-of-variables formula in measure theory:

$$\mathrm{CRPS}(L, P, Q) = \int_{\mathbb{R}^d} \mathrm{CRPS}(L_*P, L(x))dQ =$$

$$\int_{\mathbb{R}} \mathrm{CRPS}(L_*P, x)dL_*Q = \mathrm{CRPS}(L_*P, L_*Q)$$

□

Therefore, we get the easy corollary:

Corollary 7. $\mathrm{CRPS}(L, \cdot, \cdot)$ *is proper with respect to the Borel measurable sets.*

Proof. For any two probabilty distributions on the Borel measurable sets on $\mathbb{R}^d$:

$$\mathrm{CRPS}(L, Q, P) = \mathrm{CRPS}(L_*Q, L_*P) \leq$$

$$\mathrm{CRPS}(L_*P, L_*P) = \mathrm{CRPS}(L, P, P).$$

□

If $d > 1$, then for any reasonable large convex set of probability measures $\mathcal{P}'$ (e.g., the class of probabilty measures with finite first moments), and for any $P \in \mathcal{P}'$, it is always possible to find multiple Q's in $\mathcal{P}'$ such that $L_*Q = L_*P$. Therefore $d > 1$ implies that $\mathrm{CRPS}(L, \cdot, \cdot)$ is proper but not strictly proper.