Temporal Latent Auto-Encoder: A Method for Probabilistic Multivariate Time Series Forecasting

Nam Nguyen*, **Brian Quanz***
IBM Research
{nnguyen,blquanz}@us.ibm.com

Abstract

Probabilistic forecasting of high dimensional multivariate time series is a notoriously challenging task, both in terms of computational burden and distribution modeling. Most previous work either makes simple distribution assumptions or abandons modeling cross-series correlations. A promising line of work exploits scalable matrix factorization for latent-space forecasting, but is limited to linear embeddings, unable to model distributions, and not trainable end-to-end when using deep learning forecasting. We introduce a novel temporal latent auto-encoder method which enables nonlinear factorization of multivariate time series, learned end-to-end with a temporal deep learning latent space forecast model. By imposing a probabilistic latent space model, complex distributions of the input series are modeled via the decoder. Extensive experiments demonstrate that our model achieves state-of-the-art performance on many popular multivariate datasets, with gains sometimes as high as 50% for several standard metrics.

Introduction

Forecasting - predicting future values of time series, is a key component in many industries (Fildes et al. 2008). Applications include forecasting supply chain and airline demand (Fildes et al. 2008; Seeger, Salinas, and Flunkert 2016), financial prices (Kim 2003), and energy, traffic or weather patterns (Chatfield 2000). Forecasts are often required for large numbers of related time series, i.e., multivariate time series forecasting, as opposed to univariate (single time series) forecasting. For example, retailers may require sales/demand forecasts for millions of different products at thousands of different locations - amounting to billions of sales time series.

In multivariate settings, one common approach is to fit a single multi-output model to predict all series simultaneously. This includes statistical methods like vector auto-regressive (VAR) models (Lütkepohl 2005) and generalizations (e.g., MGARCH (Bauwens, Laurent, and Rombouts 2006)), and multivariate state-space models (Durbin and Koopman 2012), as well as deep neural net (DNN) models including recurrent neural networks (RNNs) (Funahashi and Nakamura 1993), temporal convolutional neural networks (TCNs) (Bai, Kolter, and Koltun 2018), and combinations (Lai et al. 2018; Goel, Melnyk, and Banerjee 2017; Borovykh, Bohte, and Oosterlee 2017; Cheng, Huang, and Zheng 2020; Dasgupta and Osogami 2017; Cirstea et al. 2018; Rodrigues and Pereira 2020). However, these are prone to overfitting and not scalable as the number of time series increases (Yu, Rao, and Dhillon 2016; Sen, Yu, and Dhillon 2019; Salinas et al. 2019).

As such, another popular approach is to abandon multivariate forecasting entirely and perform univariate forecasting (i.e., fit a separate model per series). Classical statistical forecasting methods using simple parametric models of past values and forecasts are still arguably most commonly used in industry, such as auto-regressive AR and ARIMA models (Hyndman and Athanasopoulos 2018), exponential smoothing (ES) (McKenzie 1984), and more general state-space models (Hyndman et al. 2008). Such methods have consistently out-performed machine learning methods such as RNNs in large scale forecasting competitions until recently (Makridakis, Hyndman, and Petropoulos 2020; Makridakis, Spiliotis, and Assimakopoulos 2018, 2020; Crone, Hibon, and Nikolopoulos 2011; Benidis et al. 2020). A key reason for recent success of deep learning for forecasting is multi-task univariate forecasting - sharing deep learning model parameters across all series, possibly with some series-specific scaling factors or parametric model components (Salinas, Flunkert, and Gasthaus 2019; Smyl 2020; Bandara, Bergmeir, and Hewamalage 2020; Li et al. 2019; Wen et al. 2017; Rangapuram et al. 2018; Chen et al. 2018). E.g., the winner of the M4 forecasting competition (Makridakis, Spiliotis, and Assimakopoulos 2020) was a hybrid ES-RNN model (Smyl 2020), in which a single shared univariate RNN model is used to forecast each series but seasonal and level ES parameters are simultaneously learned per series to normalize them.

However, a fundamental limitation of multi-task univariate forecasting approaches is they are unable to model cross-series correlations/effects (Rangapuram et al. 2018; Salinas et al. 2019), common in many domains (Salinas et al. 2019; Tsay 2013; Rasul et al. 2020). For example, in retail, cross-product effects (e.g., increased sales of one product causing increased/decreased sales of related products) are well known (Gelper, Wilms, and Croux 2016; Leeflang et al. 2008; Srinivasan, Ramakrishnan, and Grasman 2005). In financial time series one stock price may depend on relative prices of other stocks; and energy time series may have spatial correlations and dependencies. Furthermore, these approaches cannot

*Nam Nguyen and Brian Quanz contributed equally to this work

leverage the extra information provided from related series in case of noise or sparsity. E.g., sales are often sparse (e.g., one sale a month for a particular product and store), so the sales rate cannot be accurately estimated from a single series.

A promising line of research we focus on that addresses limitations of both the single, large multi-output multivariate model and the multi-task univariate model approaches is to use factorization (Yu, Rao, and Dhillon 2016; Sen, Yu, and Dhillon 2019). Relationships between time series are factorized into a low rank matrix, i.e., each time series is modeled as a linear combination of a smaller set of latent, basis (or *global*) time series, so forecasting can be performed in the low-dimensional latent space then mapped back to the input (*local*) space. Thus modeling can scale to very large number of series while still capturing cross-series relationships. Temporal regularized matrix factorization (TRMF) (Yu, Rao, and Dhillon 2016) imposes temporal regularization on the latent time series so they are predictable by linear auto-regressive models. Recently, DeepGLO (Sen, Yu, and Dhillon 2019) extended this approach to enable non-linear latent space forecast models. DeepGLO iteratively alternates between linear matrix factorization and fitting a latent space TCN; forecasts from this model are then fed as covariates to a separately trained multi-task univariate TCN model.

However, these have several key limitations. First, they cannot capture nonlinear relationships between series via the transformation, which are common in many domains. E.g., products' sales or stocks' prices may depend on relative price compared to others (i.e., value ratios, a non-linear relationship). Second, although deepGLO introduces deep learning, it is not an end-to-end model. Since factorization is done separately and heuristic, alternating optimization with no convergence guarantees is used, the process is inefficient and may not find an optimal solution. Third, they have no way to provide probabilistic outputs (i.e., predictive distributions), which are critical for practical use of forecasts (Makridakis, Hyndman, and Petropoulos 2020). Fourth, they are limited to capturing stationary relationships between time series with the fixed linear transform on a single time point - whereas relationships between series are likely often nonstationary.

To address these limitations and extend the factorization line of research, we propose the Temporal Latent Autoencoder (TLAE) (see Figure 1), which enables non-linear transforms of the input series trained end-to-end with a DNN temporal latent model to enforce predictable latent temporal patterns, and implicitly infers the joint predictive distribution simultaneously. Our **main contributions** are:

- We enable nonlinear factorization for the latent temporal factorization line of research; we generalize the linear mappings to/from the latent space to nonlinear transforms by replacing them with encoder and decoder neural networks, with an input-output reproduction objective, i.e., an autoencoder (Kramer 1991; Hinton and Zemel 1994). Further, the autoencoder can use temporal models (e.g., RNNs) - so embeddings can evolve over time (be nonstationary).
- We introduce temporal regularization in the latent space with flexible deep learning temporal models that can be trained end-to-end with stochastic gradient descent, by combining the objectives for reconstruction and forecast error in the latent and input spaces in the loss function.
- We enable probabilistic output sampling by injecting noise in the latent space prior to latent forecast decoding, so the model learns to implicitly model the cross-time-series joint predictive distribution by transforming the noise, similar to variational autoencoders (VAEs) (Doersch 2016; Kingma and Welling 2014a). Unlike VAEs, the latent mean (output of the latent forecast model) is not constrained.
- We perform extensive experiments with multiple multivariate forecasting datasets, demonstrating superior performance compared to past global factorization approaches as well as comparable or superior performance to other recent state of the art forecast methods, for both point and probabilistic predictions. We also provide a variety of analyses including hyper parameter sensitivity and ablation studies.

Related Work

Neural nets have a long history of applications in forecasting (Zhang, Patuwo, and Hu 1998; Benidis et al. 2020), historically mostly focused on univariate models. Here we discuss details of recent related deep learning work beyond those mentioned in the introduction. For further details on classical methods please refer to (Hyndman and Athanasopoulos 2018; Lütkepohl 2005; Durbin and Koopman 2012; Bauwens, Laurent, and Rombouts 2006). We compare with representative classical forecast methods in experiments - e.g., VAR, ARIMA, and state space models (ETS).

A trend in DNN forecasting is to normalize series to address different scaling / temporal patterns (Lai et al. 2018; Zhang 2003; Salinas, Flunkert, and Gasthaus 2019; Goel, Melnyk, and Banerjee 2017; Cheng, Huang, and Zheng 2020; Bandara, Bergmeir, and Hewamalage 2020; Smyl 2020). E.g., LSTNet (Lai et al. 2018) fits the sum of a linear AR model and a DNN with convolutional and recurrent layers. A popular multi-task univariate forecast method, DeepAR (Salinas, Flunkert, and Gasthaus 2019), scales each series by its average and fits a shared RNN across series. Another recent sate-of-the-art multi-task univariate model (Li et al. 2019) combines TCN embeddings with the Transformer architecture (Vaswani et al. 2017). Although these work well on some datasets, as mentioned they are limited in use as they cannot model dependencies between series.

TADA(Chen et al. 2018), DA-RNN (Qin et al. 2017) and GeoMAN (Liang et al. 2018) use encoder-decoder approaches built on sequence-to-sequence work (Cho et al. 2014; Bahdanau, Cho, and Bengio 2015). However the encoder-decoder is not an autoencoder, is designed for factoring in exogenous variables for multi-step univariate forecasting - not modeling cross series relationships / multivariate forecasting, and is not probabilistic. An autoencoder was used in (Cirstea et al. 2018), but only for pre-processing / denoising of individual series before training an RNN, so did not consider factorizing cross-series relationships or deriving probabilistic outcomes, as in our method.

Recently a few DNN models have also been proposed to model multivariate forecast distributions (Salinas et al. 2019; Wang et al. 2019; Rasul et al. 2020). A low-rank Gaussian copula model was proposed (Salinas et al. 2019) in which a multitask univariate LSTM (Hochreiter and Schmidhuber

1997) is used to output transformed time series and diagonal and low-rank factors of a Gaussian covariance matrix. However, it is limited in flexibility / distributions it can model, sensitive to choice of rank, and difficult to scale to very high dimensional settings. A deep factor generative model was proposed (Wang et al. 2019) in which a linear combination of RNN latent global factors plus parametric noise models the local series distributions. However, this can only model linear combinations of global series and specific noise distributions, has no easy way to map from local to global series, and is inefficient for inference and learning (limited network and data size that can be practically used). Further, a recent concurrent work uses normalizing flows for probabilistic forecasting (Rasul et al. 2020): a multivariate RNN is used to model the series progressions (single large multi-output model), with the state translated to the output joint distribution via a normalizing flow approach (Dinh, Sohl-Dickstein, and Bengio 2017).However, invertible flow requires the same number of latent dimensions as input dimensions, so it does not scale to large numbers of time series. E.g., the temporal model it is applied across all series instead of a low dimensional space as in our model, so for RNN it has quadratic complexity in the number of series, whereas ours can be much lower (shown in supplement).

Another line of related work is on variational methods with sequence models such as variational RNN (VRNN) (Chung et al. 2015) and (Chatzis 2017), e.g., VRNN applies a VAE to each hidden state of an RNN over the input series. Both of these apply the RNN over the input space so lack scalability benefits and require propagating multi-step predictions through the whole model, unlike our method which scalably applies the RNN and its propagation in a low-dimensional latent space. Further, due to noise added at every time step, VRNN may struggle with long term dependencies, and the authors state the model is designed for cases of high signal-to-noise ratio, whereas most forecast data is very noisy.

Problem Setup and Methodology

Notation A matrix of multivariate time series is denoted by a bold capital letter, univariate series by bold lowercase letters. Given a vector $\mathbf{x}$, its i-th element is denoted by x_i. For a matrix $\mathbf{X}$, we use $\mathbf{x}_i$ as the i-th column and $x_{i,j}$ is the (i,j)-th entry of $\mathbf{X}$. $\|\mathbf{X}\|_{\ell_2}$ is the matrix Frobenius norm. $\|\mathbf{x}\|_{\ell_p}$ is the ℓ_p-norm of the vector $\mathbf{x}$, defined as $(\sum_i x_i^p)^{1/p}$. Given a matrix $\mathbf{Y} \in \mathbb{R}^{n\times T}$, $\mathbf{Y}_B$ is indicated as a sub-matrix of $\mathbf{Y}$ with column indices in B. For a set $\mathcal{B}$, $|\mathcal{B}|$ is regarded as the cardinality of this set. Lastly, for functions f and g, $f \circ g$ is the composite function, $f \circ g(\mathbf{x}) = f(g(\mathbf{x}))$.

Problem definition Let a collection of high dimensional multivariate time series be denoted by $(\mathbf{y}_1, ..., \mathbf{y}_T)$, where each $\mathbf{y}_i$ at time point i is a vector of dimension n. Here we assume n is often a large number, e.g., $\sim 10^3$ to 10^6 or more. We consider the problem of forecasting τ future values $(\mathbf{y}_{T+1}, ..., \mathbf{y}_{T+\tau})$ of the series given its observed history $\{\mathbf{y}_i\}_{i=1}^T$. A more difficult but interesting problem is modeling the conditional probability distribution of the high dimensional vectors:

$$p(\mathbf{y}_{T+1}, ..., \mathbf{y}_{T+\tau} | \mathbf{y}_{1:T}) = \prod_{i=1}^{\tau} p(\mathbf{y}_{T+i} | \mathbf{y}_{1:T+i-1}). \quad (1)$$

This decomposition turns the problem of probabilistically forecasting several steps ahead to rolling prediction: the prediction at time i is input to the model to predict the value at time $i+1$. Next we describe our key contribution ideas in deterministic settings, then extend it to probabilistic ones.

Point Prediction

Motivation Temporal regularized matrix factorization (TRMF) (Yu, Rao, and Dhillon 2016), decomposes the multivariate time series represented as a matrix $\mathbf{Y} \in \mathbb{R}^{n\times T}$ (composed of n time series in its rows) into components $\mathbf{F} \in \mathbb{R}^{n\times d}$ and $\mathbf{X} \in \mathbb{R}^{d\times T}$ while also imposing temporal constraints on $\mathbf{X}$. The matrix $\mathbf{X}$ is expected to inherit temporal structures such as smoothness and seasonality of the original series. If $\mathbf{Y}$ can be reliably represented by just the few time series in $\mathbf{X}$, then tasks on the high-dimensional series $\mathbf{Y}$ can be performed on the much smaller dimensional series $\mathbf{X}$. In (Yu, Rao, and Dhillon 2016) forecasting future values of $\mathbf{Y}$ is replaced with the much simpler task of predicting future values on the latent series $\mathbf{X}$, so the $\mathbf{Y}$ prediction is just a weighted combination of the new $\mathbf{X}$ values with weights defined by the matrix $\mathbf{F}$.

To train temporal DNN models like RNNs, data is batched temporally. Denote $\mathbf{Y}_B$ as a batch of data containing a subset of b time samples, $\mathbf{Y}_B = [\mathbf{y}_t, \mathbf{y}_{t+1}, ..., \mathbf{y}_{t+b-1}]$ where $B = \{t, ..., t+b-1\}$ are time indices. To perform constrained factorization (Yu, Rao, and Dhillon 2016) proposed to solve:

$$\min_{\mathbf{X},\mathbf{F},\mathbf{W}} \mathcal{L}(\mathbf{X}, \mathbf{F}, \mathbf{W}) = \frac{1}{|\mathcal{B}|} \sum_{B\in\mathcal{B}} \mathcal{L}_B(\mathbf{X}_B, \mathbf{F}, \mathbf{W}), \quad (2)$$

where $\mathcal{B}$ is the set of all data batches and each batch loss is:

$$\mathcal{L}_B(\mathbf{X}_B, \mathbf{F}, \mathbf{W}) \triangleq \tfrac{1}{nb}\|\mathbf{Y}_B - \mathbf{F}\mathbf{X}_B\|_{\ell_2}^2 + \lambda \mathcal{R}(\mathbf{X}_B; \mathbf{W}). \quad (3)$$

Here, $\mathcal{R}(\mathbf{X}_B; \mathbf{W})$ is regularization parameterized by $\mathbf{W}$ on $\mathbf{X}_B$ to enforce certain properties of the latent factors and λ is the regularization parameter. In order to impose temporal constraints, (Yu, Rao, and Dhillon 2016) assumes an autoregressive model on $\mathbf{X}_B$ specified simply as $\mathbf{x}_\ell = \sum_{j=1}^{L} \mathbf{W}^{(j)} \mathbf{x}_{\ell-j}$ where L is a predefined lag parameter. Then, the regularization reads

$$\mathcal{R}(\mathbf{X}_B; \mathbf{W}) \triangleq \sum_{\ell=L+1}^{b} \|\mathbf{x}_\ell - \sum_{j=1}^{L} \mathbf{W}^{(j)} \mathbf{x}_{\ell-j}\|_{\ell_2}^2. \quad (4)$$

The optimization is solved via alternating minimization with respect to variables $\mathbf{X}, \mathbf{F}$, and $\mathbf{W}$.

Recently, (Sen, Yu, and Dhillon 2019) considered applying deep learning to the same problem; the authors proposed to replace the autoregressive component with a temporal convolutional network (TCN) (Bai, Kolter, and Koltun 2018). Their TCN-MF model employed the following regularization

$$\mathcal{R}(\mathbf{X}_B; \mathbf{W}) \triangleq \sum_{\ell=L+1}^{b} \|\mathbf{x}_\ell - \text{TCN}(\mathbf{x}_{\ell-L,...,\mathbf{x}_{\ell-1}}; \mathbf{W})\|_{\ell_2}^2, \quad (5)$$

where $\mathbf{W}$ is the set of parameters of the TCN network; alternating minimization was also performed for optimization.

(Sen, Yu, and Dhillon 2019) also investigated feeding TCN-MF predictions as "global" features into a "local" multi-task model forecasting individual time series. However, as mentioned, both (Yu, Rao, and Dhillon 2016) and (Sen, Yu, and Dhillon 2019) have several challenging limitations. First, due to the linear nature of the matrix factorization, the models implicitly assume linear relationships across time series. This implies the models will fail to capture non-linear correlation cross series (e.g., one series inversely proportional to another) that often occurs in practice, separately from the global temporal patterns. Second, implementation of these optimization problems with alternating minimization is sufficiently involved, especially when the loss has coupling terms as in (4). In (Sen, Yu, and Dhillon 2019), although the simple autoregressive model is replaced by a TCN, this network cannot incorporate the factorization part, making back-propagation impossible to perform end-to-end. TCN-MF model is therefore unable to leverage recent deep learning optimization developments. This may explain why solutions of TCN-MF are sometimes sub-optimal as compared to the simpler TRMF approach (Yu, Rao, and Dhillon 2016).

Our model In this paper we propose a new model to overcome these weaknesses. We observe that if $\mathbf{Y}$ can be decomposed exactly by $\mathbf{F}$ and $\mathbf{X}$, then $\mathbf{X} = \mathbf{F}^+\mathbf{Y}$ where $\mathbf{F}^+$ is the pseudo-inverse of $\mathbf{F}$. This implies that $\mathbf{Y} = \mathbf{F}\mathbf{F}^+\mathbf{Y}$.

Now if $\mathbf{F}^+$ can be replaced by an encoder and $\mathbf{F}$ by a decoder, we can exploit the ideas of autoencoders (Kramer 1991; Hinton and Zemel 1994) to seek more powerful nonlinear decompositions. The latent representation is now a nonlinear transformation of the input, $\mathbf{X} = g_{\boldsymbol{\phi}}(\mathbf{Y})$ where $g_{\boldsymbol{\phi}}$ is the encoder that maps $\mathbf{Y}$ to d dimensional $\mathbf{X}$: $g : \mathbb{R}^n \to \mathbb{R}^d$ and $\boldsymbol{\phi}$ is the set of parameters of the encoder. The nonlinearity of the encoder allows the model to represent more complex structure of the data in the latent embedding. The reconstruction of $\mathbf{Y}$ is $\hat{\mathbf{Y}} = f_{\boldsymbol{\theta}}(\mathbf{X})$ where $f_{\boldsymbol{\theta}}$ is the decoder that maps $\mathbf{X}$ back to the original domain: $f : \mathbb{R}^d \to \mathbb{R}^n$ and $\boldsymbol{\theta}$ is the set of parameters associated with the decoder.

Additionally, we introduce a new layer between the encoder and decoder to capture temporal structure of the latent representation. The main idea is illustrated in Figure 1; in the middle layer an LSTM network (Hochreiter and Schmidhuber 1997) is employed to encode the long-range dependency of the latent variables. The flow of the model is as follows: a batch of the time series $\mathbf{Y}_B = [\mathbf{y}_1, ..., \mathbf{y}_b] \in \mathbb{R}^{n\times b}$ is embedded into the latent variables $\mathbf{X}_B = [\mathbf{x}_1, ..., \mathbf{x}_b] \in \mathbb{R}^{d\times b}$ with $d \ll n$. These sequential ordered $\mathbf{x}_i$ are input to the LSTM to produce outputs $\hat{\mathbf{x}}_{L+1}, ..., \hat{\mathbf{x}}_b$ with each $\hat{\mathbf{x}}_{i+1} = h_{\mathbf{W}}(\mathbf{x}_{i-L+1}, ..., \mathbf{x}_i)$ where h is the mapping function. h is characterized by the LSTM network with parameters $\mathbf{W}$. The decoder will take the matrix $\hat{\mathbf{X}}_B$ consisting of variables $\mathbf{x}_1, ..., \mathbf{x}_L$ and $\hat{\mathbf{x}}_{L+1}, ..., \hat{\mathbf{x}}_b$ as input and yield the matrix $\hat{\mathbf{Y}}_B$.

As seen from the figure, batch output $\hat{\mathbf{Y}}_B$ contains two components. The first, $\hat{\mathbf{y}}_i$ with $i = 1, ..., L$, is directly transferred from the encoder without passing through the middle layer: $\hat{\mathbf{y}}_i = f_{\boldsymbol{\theta}} \circ g_{\boldsymbol{\phi}}(\mathbf{y}_i)$, while the second component $\hat{\mathbf{y}}_i$ with $i = L+1, ..., b$ is a function of the past input: $\hat{\mathbf{y}}_{i+1} = f_{\boldsymbol{\theta}} \circ h_{\mathbf{W}} \circ g_{\boldsymbol{\phi}}(\mathbf{y}_{i-L+1}, ..., \mathbf{y}_i)$. By minimizing the error $\|\hat{\mathbf{y}}_i - \mathbf{y}_i\|_{\ell_p}^p$, one can think of this second component as providing the model the capability to predict the future from the observed history, while at the same time the first one requires the model to reconstruct the data faithfully.

The objective function with respect to a batch of data is defined as follows.

$$\begin{aligned}\mathcal{L}_B(\mathbf{W}, \boldsymbol{\phi}, \boldsymbol{\theta}) &\triangleq \tfrac{1}{nb}\|\mathbf{Y}_B - \hat{\mathbf{Y}}_B\|_{\ell_p}^p \\ &+ \lambda \tfrac{1}{d(b-L)} \textstyle\sum_{i=L}^{b-1} \|\mathbf{x}_{i+1} - h_{\mathbf{W}}(\mathbf{x}_{i-L+1}, ..., \mathbf{x}_i)\|_{\ell_q}^q,\end{aligned} \tag{6}$$

and the overall loss function is

$$\min_{\mathbf{W},\boldsymbol{\phi},\boldsymbol{\theta}} \mathcal{L}(\mathbf{W}, \boldsymbol{\phi}, \boldsymbol{\theta}) = \tfrac{1}{|\mathcal{B}|} \textstyle\sum_{B\in\mathcal{B}} \mathcal{L}_B(\mathbf{W}, \boldsymbol{\phi}, \boldsymbol{\theta}). \tag{7}$$

On the one hand, by optimizing $\hat{\mathbf{Y}}$ to be close to $\mathbf{Y}$, the model is expected to capture the correlation cross time series and encode this global information into a few latent variables $\mathbf{X}$. On the other hand, minimizing the discrepancy between $\hat{\mathbf{X}}$ and $\mathbf{X}$ allows the model to capture temporal dependency and provide the predictive capability of the latent representation. We add a few more remarks:

- Although we use LSTMs here, other networks (e.g., TCN (Bai, Kolter, and Koltun 2018) or Transformer (Vaswani et al. 2017)) can be applied.
- A fundamental difference between TRMF / TCN-MF and our method is that in the former, latent variables are part of the optimization and solved for explicitly while in ours, latent variables are parameterized by the networks, thus back-propagation can be executed end-to-end for training.
- By simpler optimization, our model allows more flexible selection of loss types imposed on $\hat{\mathbf{Y}}$ and $\hat{\mathbf{X}}$. In experiments, we found that imposing ℓ_1 loss on $\hat{\mathbf{Y}}$ consistently led to better prediction while performance remains similar with either ℓ_1 or ℓ_2 loss on $\hat{\mathbf{X}}$. Since ℓ_1 loss is known to be more robust to outliers, imposing it directly on $\hat{\mathbf{Y}}$ makes the model more resilient to potential outliers.
- Encoders/decoders themselves can use temporal DNNs so non-static relationships can be captured.

Once the model is learned, forecasting several steps ahead is performed via rolling windows. Given past input data $[\mathbf{y}_{T-L+1}, ..., \mathbf{y}_T]$, the learned model produces the latent prediction $\hat{\mathbf{x}}_{T+1} = h_{\mathbf{W}}(\mathbf{x}_{T-L+1}, ..., \mathbf{x}_T)$ where each $\mathbf{x}_i = g_{\boldsymbol{\phi}}(\mathbf{y}_i)$. The predicted $\hat{\mathbf{y}}_{T+1}$ is then decoded from $\hat{\mathbf{x}}_{T+1}$. The same procedure can be sequentially repeated τ times (in the latent space) to forecast τ future values of $\mathbf{Y}$ in which the latent prediction $\hat{\mathbf{x}}_{T+2}$ utilizes $[\mathbf{x}_{T-L+1}, ..., \mathbf{x}_T, \hat{\mathbf{x}}_{T+1}]$ as the input to the model. Notice that the model does not require retraining during prediction as opposed to TRMF.

Probabilistic Prediction

One of the notorious challenges with high-dimensional time series forecasting is how to probabilistically model the future values conditioned on the observed sequence: $p(\mathbf{y}_{T+1}, ..., \mathbf{y}_{T+\tau}|\mathbf{y}_1, ..., \mathbf{y}_T)$. Most previous works either focus on modelling each individual time series or parameterizing the conditional probability of the high dimensional series by a multivariate Gaussian distribution. However, this is inconvenient since the number of learned parameters (covariance matrix) grows quadratically with the data dimension. Recent DNN approaches make distribution assumptions

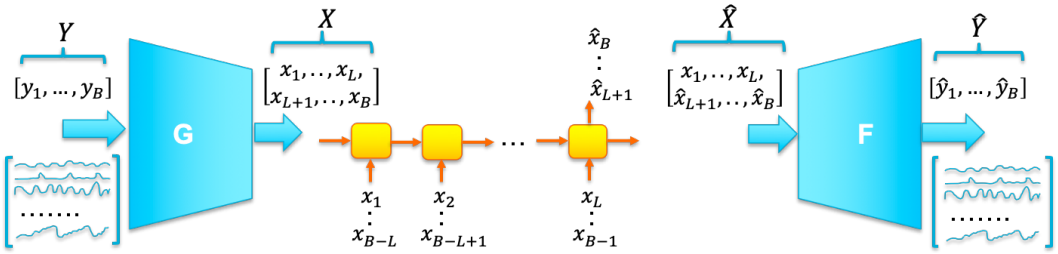

Figure 1: Temporal latent autoencoder. Though illustrated with an RNN, any temporal DNN model (e.g., TCN or Transformer) can be used in the latent space. The decoder translates Normal noise to arbitrary distributions.

(such as low-rank covariance) that limit flexibility and/or similarly lack scalability (see Related Work section).

In this paper, instead of directly modelling in the input space, we propose to encode the high dimensional data to a much lower dimensional embedding, on which a probabilistic model can be imposed. Prediction samples are later obtained by sampling from the latent distribution and translating these samples through the decoder. If the encoder is sufficiently complex so that it can capture non-linear correlation among series, we can introduce fairly simple probabilistic structure on the latent variables and are still able to model complex distributions of the multivariate data via the decoder mapping. Indeed, together with the proposed network architecture in Figure 1, we model

$$p(\mathbf{x}_{i+1}|\mathbf{x}_1, ..., \mathbf{x}_i) = \mathcal{N}(\mathbf{x}_{i+1}; \boldsymbol{\mu}_i, \boldsymbol{\sigma}_i^2), \quad i = L, ..., b. \quad (8)$$

Here, we fix the conditional distribution of latent variables to multivariate Gaussian with diagonal covariance matrix with variance $\boldsymbol{\sigma}_i^2$. This is meaningful as it encourages the latent variables to capture different orthogonal patterns of the data, which makes the representation more powerful, universal, and interpretable. The mean $\boldsymbol{\mu}_i$ and variance $\boldsymbol{\sigma}_i^2$ are functions of $\mathbf{x}_1, ..., \mathbf{x}_i$: $\boldsymbol{\mu}_i = h_{\mathbf{W}}^{(1)}(\mathbf{x}_1, ..., \mathbf{x}_i)$ and $\boldsymbol{\sigma}_i^2 = h_{\mathbf{W}}^{(2)}(\mathbf{x}_1, ..., \mathbf{x}_i)$.

The objective function $\mathcal{L}_B(\phi, \theta, \mathbf{W})$ with respect to the batch data $\mathbf{Y}_B$ is defined as the weighted combination of the reconstruction loss and the negative log likelihood loss

$$\frac{\|\hat{\mathbf{Y}}_B - \mathbf{Y}_B\|_{\ell_p}^p}{nb} - \frac{\lambda}{b-L} \sum_{i=L+1}^{b} \log \mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_{i-1}, \boldsymbol{\sigma}_{i-1}^2). \quad (9)$$

This bears similarity to the loss of the variational autoencoder (VAE) (Kingma and Welling 2014b) which consists of a data reconstruction loss and a Kullback–Leibler divergence loss encouraging the latent distribution to be close to the standard multivariate Gaussian with zero mean and unit diagonal covariance. Unlike VAEs, our model has a temporal model in the latent space and is measuring a conditional discrepancy (with no fixed mean constraint). Further, rather than encourage unit variance we fix latent space unit variance, to also help avoid overfitting during training - i.e., we set $\boldsymbol{\sigma}_i^2 = 1$ in our model. As with GANs, the decoder learns to translate this noise to arbitrary distributions (examples in supplement).

Recall the output of the decoder $\hat{\mathbf{y}}_{i+1} = f_{\boldsymbol{\theta}}(\hat{\mathbf{x}}_{i+1})$ for each sample $\hat{\mathbf{x}}_{i+1}$ from the distribution (8). In order to back-propagate through batch data, we utilize the reparameterization trick as in VAEs. I.e., $\hat{\mathbf{x}}_{i+1} = \boldsymbol{\mu}_i + \mathbf{1}\epsilon = h_{\mathbf{W}}^{(1)}(\mathbf{x}_1, ..., \mathbf{x}_i) + \mathbf{1}\epsilon$ with $\epsilon \sim \mathcal{N}(0, 1)$. Each iteration when gradient calculation is required, a sample ϵ is generated which yields latent sample $\hat{\mathbf{x}}_{i+1}$ and associated $\hat{\mathbf{y}}_{i+1}$.

Once the model is learned, next prediction samples $\hat{\mathbf{y}}_{T+1}$ can be decoded from samples $\hat{\mathbf{x}}_{T+1}$ of the latent distribution (8). Conditioned on $\mathbf{x}_{T-L+1}, ..., \mathbf{x}_T$ and the mean of $\hat{\mathbf{x}}_{T+1}$, samples $\hat{\mathbf{x}}_{T+2}$ can also be drawn from (8) to obtain prediction samples $\hat{\mathbf{y}}_{T+2}$, and so on.

Experiments

Point Estimation

We first evaluate our point prediction model with loss defined in (7). We compare with state-of-the art multivariate and univariate forecast methods (Sen, Yu, and Dhillon 2019) (Yu, Rao, and Dhillon 2016) (Salinas, Flunkert, and Gasthaus 2019) using 3 popular datasets: *traffic*: hourly traffic of 963 San Fancisco car lanes (Cuturi 2011; Dua and Graff 2017), *electricity*: hourly consumption of 370 houses (Trindad 2015), and *wiki*: daily views of ˜115k Wikipedia articles (Kaggle 2017). Traffic and electricity show weekly cross-series patterns; wiki contains a very large number of series. Following conventional setups (Salinas, Flunkert, and Gasthaus 2019; Sen, Yu, and Dhillon 2019; Yu, Rao, and Dhillon 2016), we perform rolling prediction evaluation: 24 time-points per window, last 7 windows for testing for traffic and electricity, and 14 per window with last 4 windows for wiki. We use the last few windows prior to the test period for any hyper parameter selection. We use 3 standard metrics: mean absolute percent error (MAPE), weighted absolute percent error (WAPE), and symmetric MAPE (SMAPE) to measure test prediction error. Dataset / formula details are in the supplement.

Network architecture and optimization setup in experiments is as follows: the encoder and decoder use feed forward network (FNN) layers with ReLU activations on all but the last layer. Layer dimensions vary per dataset. The network architecture in the latent space is a 4-layer LSTM, each with 32 hidden units. In all experiments, ℓ_1 loss is used on $\hat{\mathbf{Y}}$ and ℓ_2 for the regularization. Regularization parameter λ is set to 0.5. We find the ℓ_1 loss on $\hat{\mathbf{Y}}$ can help reduce potential outlier effects and provide more stable and accurate results. Setup and training details are provided in the supplement.

Table 1 shows the comparison of different approaches. All results except our proposed TLAE were reported in (Sen, Yu, and Dhillon 2019) under the same experimental setup;

we pick the best reported results in (Sen, Yu, and Dhillon 2019) with or without data normalization. Here, global models use global features for multivariate forecasting while local models employ univariate models and separately predict individual series. Here we do not compare our model with conventional methods (e.g., VAR, ARIMA) since they are already confirmed to obtain inferior performance to TRMF and DeepAR methods (Yu, Rao, and Dhillon 2016) (Salinas, Flunkert, and Gasthaus 2019).

As seen in the table, our method significantly out-performs other global modeling / factorization methods on all datasets (8/9 dataset-metric combinations) - showing it clearly advances the state-of-the-art for global factorization multivariate forecasting approaches. Compared with other global models, the gain on traffic and electricity datasets can be as significant as 50%. Further, even without any local modeling and additional exogenous features like hour of day (as used in local and combined methods), our method still achieves superior performance on 2/3 datasets across all metrics. Our model could likely be further improved by incorporating the exogenous features in the latent space or with local modeling (as done with deepGLO) - the point is our model provides a better global fit starting point. Also note our model only applied standard network architectures and did not make use of recent advanced ones such as TCNs or Transformer, for which we might expect further improvement.

Furthermore, in experiments latent dimensions are set to 16 for traffic and 32 for electricity data, as opposed to 64 dimensions used in (Sen, Yu, and Dhillon 2019). This indicates our model is able to learn a better and more compact representation. We show examples of learned latent series and input and latent space predictions (and distributions) in the supplement, illustrating our model is able to capture shared global patterns. We also highlight that our model does not need retraining during testing.

Probabilistic Estimation

Our next experiments consider probabilistic forecasting. We compare our model with the state-of-the-art probabilistic multivariate method (Salinas et al. 2019), as well as (Wang et al. 2019) and univariate forecasting (Salinas, Flunkert, and Gasthaus 2019; Rangapuram et al. 2018; Li et al. 2019) in the supplement, each following the same data setup (note: different data splits and processing than in the previous section; details in supplement). We apply the same network architecture as in previous experiments, except the latent variable loss is the negative Gaussian log likelihood (9) and the regularization parameter λ is set to 0.005. A smaller λ is selected in this case to account for the scale difference between the regularizations in (6) and (9). Latent samples are generated during training with the reparameterization trick and distribution statistics obtained from decoded sampled latent predictions. Two additional datasets are included: *solar* (hourly production from 137 stations) and *taxi* (rides taken at 1214 locations every 30 minutes) (details in the supplement).

For probabilistic estimates, we report both the continuous ranked probability score across summed time series (CRPS-sum) (Matheson and Winkler 1976; Gneiting and Raftery 2007; Salinas et al. 2019) (details in supplement) and MSE (mean square error) error metrics, to measure overall joint distribution pattern fit and fit of joint distribution central tendency, respectively, so that together the two metrics give a good idea of how good the predictive distribution fit is. Results are shown in Table 2 comparing error scores of TLAE with other methods reported in (Salinas et al. 2019). Here, GP is the Gaussian process model of (Salinas et al. 2019). As one can observe, our model outperforms other methods on most of the dataset-metric combinations (7/10), in which the performance gain is significant on Solar, Traffic, and Taxi datasets. We also provide additional tables in the supplement to show CRPS and MSE scores with standard deviation from different runs for more thorough comparison. In the supplement, we visually show different latent series learned from the model on all datasets as well as predictive distributions and sampled 2D joint distributions, demonstrating non-Gaussian and non-stationary distribution patterns. From the plots we see that some are focused on capturing global, more slowly changing patterns across time series; others appear to capture local, faster changing information. Combinations of these enable the model to provide faithful predictions.

Hyper Param. Sensitivity & Ablation Study

We conducted experiments with traffic data to monitor prediction performance when varying hyper parameters: batch size, regularization parameter λ, and latent dimension. We use the same network architecture as the previous section and train the model with probabilistic loss (9). Predictions are the decoded mean of the latent distribution and error is measured by MAPE, WAPE, and SMAPE metrics.

As explained (see Figure 1), the latent sequence to the decoder consists of two sub-sequences $\{\mathbf{x}_1, ..., \mathbf{x}_L\}$ and $\{\hat{\mathbf{x}}_{L+1}, ..., \hat{\mathbf{x}}_b\}$. The first is directly transmitted from the encoder; the second is the output of the LSTM network. Minimizing discrepancy between $\hat{\mathbf{x}}_{L+i}$ and $\mathbf{x}_{L+i}$ equips the latent variables with predictive capability, which implicitly contributes to better time series prediction, so selecting the batch size sufficiently larger than L (the number of LSTM time steps) should lead to better predictive performance.

We validate this intuition by varying batch size $b = L + 1, 1.5L, 2L, 2.5L$, and $3L$ where L is set to 194. Figure 2 illustrates the metric variability with increasing batch size. All three metrics decrease as we increase the batch size, confirming the importance of balancing the two latent sub-sequences, i.e., having a balance between both a direct reproduction and predictive loss component in the input space.

Next, for batch size $b = 2L$, we vary λ = 1e-6, 1e-5, 1e-4, 1e-3, 5e-3, 5e-2, 5e-1, and 5 and train for 500 epochs. Figure 3 shows metrics vs. λ. As latent space regularization is ignored with very small λ, the overall prediction performance is poor. The performance is quickly improved with higher λ - the latent constraint starts to dominate the reconstruction term with larger λ. The best range of λ is [1e-4, 1e-2].

Lastly, we vary the latent embedding dimension between $[2, 4, 8, 16, 32]$, and train with 200 epochs vs. 1000 to reduce computational time. Figure 4 shows the impact on metrics. The model performance slightly improves with increase of the latent dimension and starts to stabilize, indicating higher latent dimension may not help much.

Model	Algorithm	Datasets		
		Traffic	Electricity-Large	Wiki-Large
Global factorization	**TLAE** (our proposed method)	$\mathbf{0.117}^*/\mathbf{0.137}^*/\mathbf{0.108}^*$	$\mathbf{0.080}^*/\mathbf{0.152}^*/\mathbf{0.120}^*$	$0.334/\mathbf{0.447}/\mathbf{0.434}$
	DeepGLO - TCN-MF	$0.226/0.284/0.247$	$0.106/0.525/0.188$	$0.433/1.59/0.686$
	TRMF	$0.159/0.226/0.181$	$0.104/0.280/0.151$	$\mathbf{0.309}/0.847/0.451$
	SVD+TCN	$0.329/0.687/0.340$	$0.219/0.437/0.238$	$0.639/2.000/0.893$
Local & combined	DeepGLO - combined	$0.148/0.168/0.142$	$0.082/0.341/0.121$	$0.237/0.441/0.395$
	LSTM	$0.270/0.357/0.263$	$0.109/0.264/0.154$	$0.789/0.686/0.493$
	DeepAR	$0.140/0.201/0.114$	$0.086/0.259/0.141$	$0.429/2.980/0.424$
	TCN (no LeveledInit)	$0.204/0.284/0.236$	$0.147/0.476/0.156$	$0.511/0.884/0.509$
	TCN (LeveledInit)	$0.157/0.201/0.156$	$0.092/0.237/0.126$	$0.212^*/0.316^*/0.296^*$
	Prophet (Taylor and Letham 2018)	$0.303/0.559/0.403$	$0.197/0.393/0.221$	-

Table 1: Comparison of different algorithms with WAPE/MAPE/SMAPE metrics. Only local and combined models employ additional features such as hour of day and day of week. Best results for global modeling methods are labeled in bold, best overall with *. Our scores are averaged over 3 separate random initialized runs. Standard dev. is less than 0.003 for all the metrics.

	CRPS-Sum / MSE				
Estimator	Solar	Electricity-Small	Traffic	Taxi	Wiki-Small
VAR	0.524 / 7.0e3	0.031 / 1.2e7	0.144 / 5.1e-3	0.292 / -	3.400 / -
GARCH	0.869 / 3.5e3	0.278 / 1.2e6	0.368 / 3.3e-3	- / -	- / -
Vec-LSTM-ind	0.470 / 9.9e2	0.731 / 2.6e7	0.110 / 6.5e-4	0.429 / 5.2e1	0.801 / 5.2e7
Vec-LSTM-ind-scaling	0.391 / 9.3e2	0.025 / 2.1e5	0.087 / 6.3e-4	0.506 / 7.3e1	0.113 / 7.2e7
Vec-LSTM-fullrank	0.956 / 3.8e3	0.999 / 2.7e7	-/-	-/-	-/-
Vec-LSTM-fullrank-scaling	0.920 /3.8e3	0.747 / 3.2e7	-/-	-/-	-/-
Vec-LSTM-lowrank-Copula	0.319 / 2.9e3	0.064 / 5.5e6	0.103 / 1.5e-3	0.4326 / 5.1e1	0.241 / 3.8e7
LSTM-GP (Salinas et al. 2019)	0.828 / 3.7e3	0.947 / 2.7e7	2.198 / 5.1e-1	0.425 / 5.9e1	0.933 / 5.4e7
LSTM-GP-scaling (Salinas et al. 2019)	0.368 / 1.1e3	**0.022 / 1.8e5**	0.079 / 5.2e-4	0.183 / 2.7e1	1.483 / 5.5e7
LSTM-GP-Copula (Salinas et al. 2019)	0.337 / 9.8e2	0.024 / 2.4e5	0.078 / 6.9e-4	0.208 / 3.1e1	**0.086** / 4.0e7
VRNN (Chung et al. 2015)	0.133 / 7.3e2	0.051 / 2.7e5	0.181 / 8.7e-4	0.139 / 3.0e1	0.396 / 4.5e7
TLAE (our proposed method)	**0.124 / 6.8e2**	0.040 / 2.0e5	**0.069 / 4.4e-4**	**0.130 / 2.6e1**	0.241 / **3.8e7**

Table 2: Comparison of algorithms with CRPS-Sum and MSE metrics (lower is better) - mean from 3 separate random initialized runs. Most results are from Tables 2 and 6 of (Salinas et al. 2019) with VRNN and our results (TLAE) under the same setup. VAR and GARCH are traditional multivariate methods (Lütkepohl 2005; Bauwens, Laurent, and Rombouts 2006); Vec-LSTM methods use a single global LSTM that takes and predicts all series at once, with different output Gaussian distribution approaches; and GP methods are DNN gaussian process ones proposed in (Salinas et al. 2019) with GP-Copula the main one - details in (Salinas et al. 2019). A '-' indicates a method failed (e.g., required too much memory as not scalable enough for data size).

Additionally, to validate the hypothesis that nonlinear transformation helps, we performed ablation study by using a linear decoder and encoder under the same setup. We found worse performance than the non-linear case, though still better than DeepGLO (details in supplement).

Conclusion

This paper introduces an effective method for high dimensional multivariate time series forecasting, advancing the state-of-the-art for global factorization approaches. The method offers an efficient combination between flexible non-linear autoencoder mapping and inherent latent temporal dynamics modeled by an LSTM. The proposed formulation allows end-to-end training and, by modelling the distribution in the latent embedding, generating complex predictive distributions via the non-linear decoder. Our experiments illustrate the superior performance compared to other state-of-the-art methods on several common time series datasets. Future directions include testing temporal models in the autoencoder, $3D$ tensor inputs, and combinations with local modeling.

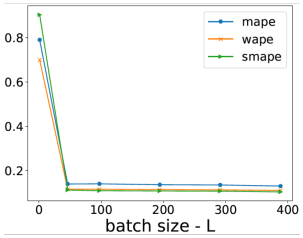

Figure 2: Vary batch size

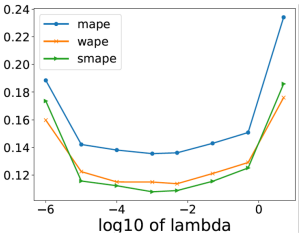

Figure 3: Vary λ

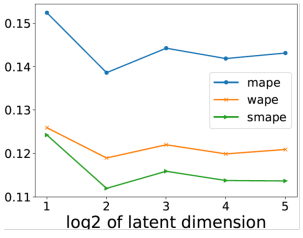

Figure 4: Vary latent dim.

References

Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*.

Bai, S.; Kolter, J. Z.; and Koltun, V. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271* .

Bandara, K.; Bergmeir, C.; and Hewamalage, H. 2020. LSTM-MSNet: Leveraging Forecasts on Sets of Related Time Series With Multiple Seasonal Patterns. *IEEE Transactions on Neural Networks and Learning Systems* 1–14.

Bauwens, L.; Laurent, S.; and Rombouts, J. V. 2006. Multivariate GARCH models: a survey. *Journal of applied econometrics* 21(1): 79–109.

Benidis, K.; Rangapuram, S. S.; Flunkert, V.; Wang, B.; Maddix, D.; Turkmen, C.; Gasthaus, J.; Bohlke-Schneider, M.; Salinas, D.; Stella, L.; et al. 2020. Neural forecasting: Introduction and literature overview. *arXiv preprint arXiv:2004.10240* .

Borovykh, A.; Bohte, S.; and Oosterlee, C. W. 2017. Conditional time series forecasting with convolutional neural networks. *arXiv preprint arXiv:1703.04691* .

Chatfield, C. 2000. *Time-series forecasting*. CRC press.

Chatzis, S. P. 2017. Recurrent latent variable conditional heteroscedasticity. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2711–2715. IEEE.

Chen, T.; Yin, H.; Chen, H.; Wu, L.; Wang, H.; Zhou, X.; and Li, X. 2018. TADA: Trend Alignment with Dual-Attention Multi-task Recurrent Neural Networks for Sales Prediction. In *2018 IEEE International Conference on Data Mining (ICDM)*, 49–58.

Cheng, J.; Huang, K.; and Zheng, Z. 2020. Towards Better Forecasting by Fusing Near and Distant Future Visions. In *AAAI Conference on Artificial Intelligence*.

Cho, K.; van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1724–1734.

Chung, J.; Kastner, K.; Dinh, L.; Goel, K.; Courville, A. C.; and Bengio, Y. 2015. A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*, 2980–2988.

Cirstea, R.-G.; Micu, D.-V.; Muresan, G.-M.; Guo, C.; and Yang, B. 2018. Correlated time series forecasting using multi-task deep neural networks. In *Proceedings of the 27th acm international conference on information and knowledge management*, 1527–1530.

Crone, S. F.; Hibon, M.; and Nikolopoulos, K. 2011. Advances in forecasting with neural networks? Empirical evidence from the NN3 competition on time series prediction. *International Journal of forecasting* 27(3): 635–660.

Cuturi, M. 2011. Fast global alignment kernels. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, 929–936.

Dasgupta, S.; and Osogami, T. 2017. Nonlinear dynamic Boltzmann machines for time-series prediction. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Dinh, L.; Sohl-Dickstein, J.; and Bengio, S. 2017. Density estimation using real nvp .

Doersch, C. 2016. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908* .

Dua, D.; and Graff, C. 2017. UCI Machine Learning Repository. URL http://archive.ics.uci.edu/ml.

Durbin, J.; and Koopman, S. J. 2012. *Time series analysis by state space methods*. Oxford university press.

Fildes, R.; Nikolopoulos, K.; Crone, S. F.; and Syntetos, A. A. 2008. Forecasting and operational research: a review. *Journal of the Operational Research Society* 59(9): 1150–1172.

Funahashi, K.-i.; and Nakamura, Y. 1993. Approximation of dynamical systems by continuous time recurrent neural networks. *Neural networks* 6(6): 801–806.

Gelper, S.; Wilms, I.; and Croux, C. 2016. Identifying Demand Effects in a Large Network of Product Categories. *Journal of Retailing* 92(1): 25 – 39. ISSN 0022-4359. doi:https://doi.org/10.1016/j.jretai.2015.05.005. URL http://www.sciencedirect.com/science/article/pii/S0022435915000536.

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

Goel, H.; Melnyk, I.; and Banerjee, A. 2017. R2N2: residual recurrent neural networks for multivariate time series forecasting. *arXiv preprint arXiv:1709.03159* .

Hinton, G. E.; and Zemel, R. S. 1994. Autoencoders, minimum description length and Helmholtz free energy. In *Advances in neural information processing systems*, 3–10.

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

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

Hyndman, R. J.; and Athanasopoulos, G. 2018. *Forecasting: principles and practice*. OTexts.

Kaggle. 2017. Wikipedia web traffic data set. URL https://www.kaggle.com/c/web-traffic-time-series-forecasting/data. Accessed: 2020-01-11.

Kim, K.-j. 2003. Financial time series forecasting using support vector machines. *Neurocomputing* 55(1-2): 307–319.

Kingma, D. P.; and Welling, M. 2014a. Auto-Encoding Variational Bayes. In Bengio, Y.; and LeCun, Y., eds., *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. URL http://arxiv.org/abs/1312.6114.

Kingma, D. P.; and Welling, M. 2014b. Auto-encoding variational bayes. In *In Proceedings of the International Conference on Learning Representations (ICLR)*.

Kramer, M. A. 1991. Nonlinear principal component analysis using autoassociative neural networks. *AIChE journal* 37(2): 233–243.

Lai, G.; Chang, W.-C.; Yang, Y.; and Liu, H. 2018. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, 95–104.

Leeflang, P. S.; Selva], J. P.; Dijk], A. V.; and Wittink, D. R. 2008. Decomposing the sales promotion bump accounting for cross-category effects. *International Journal of Research in Marketing* 25(3): 201 – 214. ISSN 0167-8116. doi:https://doi.org/10.1016/j.ijresmar.2008.03.003. URL http://www.sciencedirect.com/science/article/pii/S0167811608000347.

Li, S.; Jin, X.; Xuan, Y.; Zhou, X.; Chen, W.; Wang, Y.-X.; and Yan, X. 2019. Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting. In *Advances in neural information processing systems*, 4838–4847.

Liang, Y.; Ke, S.; Zhang, J.; Yi, X.; and Zheng, Y. 2018. Geoman: Multi-level attention networks for geo-sensory time series prediction. In *IJCAI*, 3428–3434.

Lütkepohl, H. 2005. *New introduction to multiple time series analysis*. Springer Science & Business Media.

Makridakis, S.; Hyndman, R. J.; and Petropoulos, F. 2020. Forecasting in social settings: The state of the art. *International Journal of Forecasting* 36(1): 15–28.

Makridakis, S.; Spiliotis, E.; and Assimakopoulos, V. 2018. Statistical and Machine Learning forecasting methods: Concerns and ways forward. *PloS one* 13(3).

Makridakis, S.; Spiliotis, E.; and Assimakopoulos, V. 2020. The M4 Competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting* 36(1): 54–74.

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

McKenzie, E. 1984. General exponential smoothing and the equivalent ARMA process. *Journal of Forecasting* 3(3): 333–344.

Qin, Y.; Song, D.; Cheng, H.; Cheng, W.; Jiang, G.; and Cottrell, G. W. 2017. A dual-stage attention-based recurrent neural network for time series prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 2627–2633.

Rangapuram, S. S.; Seeger, M.; Gasthaus, J.; Stella, L.; Wang, Y.; and Januschowski, T. 2018. Deep State Space Models for Time Series Forecasting. In *Advances in neural information processing systems*, 7796—7805.

Rasul, K.; Sheikh, A.-S.; Schuster, I.; Bergmann, U.; and Vollgraf, R. 2020. Multi-variate Probabilistic Time Series Forecasting via Conditioned Normalizing Flows. *arXiv preprint arXiv:2002.06103* .

Rodrigues, F.; and Pereira, F. C. 2020. Beyond expectation: Deep joint mean and quantile regression for spatiotemporal problems. *IEEE Transactions on Neural Networks and Learning Systems* .

Salinas, D.; Bohlke-Schneider, M.; Callot, L.; Medico, R.; and Gasthaus, J. 2019. High-Dimensional Multivariate Forecasting with Low-Rank Gaussian Copula Processes. In *Advances in neural information processing systems*, 7796––7805.

Salinas, D.; Flunkert, V.; and Gasthaus, J. 2019. DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks. *International Journal of Forecasting* 8(2): 136–153.

Seeger, M. W.; Salinas, D.; and Flunkert, V. 2016. Bayesian intermittent demand forecasting for large inventories. In *Advances in Neural Information Processing Systems*, 4646–4654.

Sen, R.; Yu, H.-F.; and Dhillon, I. 2019. Think Globally, Act Locally: A Deep Neural Network Approach to High-Dimensional Time Series Forecasting. In *Advances in neural information processing systems*, 4838–4847.

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

Srinivasan, S. R.; Ramakrishnan, S.; and Grasman, S. E. 2005. Identifying the effects of cannibalization on the product portfolio. *Marketing intelligence & planning* .

Taylor, S. J.; and Letham, B. 2018. Forecasting at scale. *The American Statistician* 72(1): 37–45.

Trindad, A. 2015. ElectricityLoadDiagrams20112014 Data Set. URL https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014. Accessed: 2020-01-11.

Tsay, R. S. 2013. *Multivariate time series analysis: with R and financial applications*. John Wiley & Sons.

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

Wang, Y.; Smola, A.; Maddix, D.; Gasthaus, J.; Foster, D.; and Januschowski, T. 2019. Deep Factors for Forecasting. In *International Conference on Machine Learning*, 6607–6617.

Wen, R.; Torkkola, K.; Narayanaswamy, B.; and Madeka, D. 2017. A multi-horizon quantile recurrent forecaster. In *Advances in neural information processing systems - Time Series Workshop*.

Yu, H.-F.; Rao, N.; and Dhillon, I. 2016. Temporal Regularized Matrix Factorization for High-dimensional Time Series Prediction. In *Advances in neural information processing systems*, 847–855.

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

Zhang, G. P. 2003. Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing* 50: 159–175.