FR^{3}LS: a Forecasting model with Robust and Reduced Redundancy Latent Series

Abdallah Aaraba[1](✉), Shengrui Wang[1], and Jean-Marc Patenaude[2]

[1] University of Sherbrooke, Sherbrooke, QC, Canada
{abdallah.aaraba,shengrui.wang}@usherbrooke.ca
[2] Laplace Insights, Sherbrooke, QC, Canada
jeanmarc@laplaceinsights.com

Abstract. While some methods are confined to linear embeddings and others exhibit limited robustness, high-dimensional time series factorization techniques employ scalable matrix factorization for forecasting in latent space. This paper introduces a novel factorization method that employs a non-contrastive approach, guiding an autoencoder-like architecture to extract robust latent series while minimizing redundant information within the embeddings. The resulting learned representations are utilized by a temporal forecasting model, generating forecasts within the latent space, which are subsequently decoded back to the original space through the decoder. Extensive experiments demonstrate that our model achieves state-of-the-art performance on numerous commonly used datasets.

Keywords: Time series factorization · Probabilistic forecasting · Non-contrastive learning.

1 Introduction

Modern time series forecasting, involving correlated multivariate time series over an extended period, encounters challenges with conventional methods like autoregressive models (AR, ARIMA) [12], especially when handling large datasets with hundreds of thousands of time series due to scalability issues. Deep learning, exemplified by LSTM [8] and Temporal Convolution Networks (TCN) [1], addresses this by training on the entire dataset, utilizing shared model parameters. However, these deep learning methods inherently struggle with capturing inter-series interactions and correlations observed in diverse domains [24].

A promising research direction explores factorizing time series relationships into a low-rank matrix, yielding a concise latent time series representation [22, 27, 32]. Temporal Regularized Matrix Factorization (TRMF) [32] achieves this by representing each time series with a linear combination of a few latent series and applying linear temporal regularization to ensure temporal dependencies. Forecasted values in the latent space are transformed back using matrix multiplication. DeepGLO [27] extends TRMF with nonlinear regularization, incorporating iterative training between linear matrix factorization and fitting a latent space Temporal Convolutional Network (TCN). Another advancement, named Temporal Latent AutoEncoder (TLAE) [22], incorporates a nonlinear framework through an autoencoder to extract the latent time series space. TLAE forecasts within the embedding space, and then transforms forecast to the original space through a decoder. All these factorization methods lack robustness, overlooking random noise and distortions in data that may lead to issues like overfitting.

To address the aforementioned limitations and extend the existing line of factorization research, we introduce the FR^{3}LS forecasting model. Illustrated in Figure 1, FR^{3}LS nonlinearly projects high-dimensional time series into a latent space with manageable dimensions, facilitating future value forecasting using latent representations. The final predictions are derived by decoding latent forecasts generated by the middle layer forecasting model. This latent prediction approach serves to regularize the embedding space (latent space), capturing temporal dependencies between embeddings. Furthermore, latent representations undergo additional regularization through a non-contrastive objective with only positive samples. This means that the model is trained to produce embeddings robust to distortions applied to input subseries (i.e., augmented context views), while minimizing redundancy between components of the vector embedding. Thus, we present an all-in-one model capable of learning robust representations while maintaining a connection between forecasts in latent and original spaces.

2 Related Work

We focus on recent deep learning approaches beyond traditional methods. Further details on classical methods (e.g., AR and ARIMA) can be found in [2, 12, 20]. Deep learning methods, encompassing RNNs [23, 26], CNNs [1], GNNs (Graph NNs) [4], and Transformers [36], have gained acclaim for their effectiveness in time series forecasting, surpassing classical models like ARIMA and VAR (Vector AR). For instance, TCN [1] uses dilated convolutions to enhance efficiency and predictive performance over traditional RNNs. Models like LSTnet [15] combine CNNs and RNNs to capture short-term local dependencies and long-term trends. Additionally, LogTrans [16] and Informer [36] address self-attention efficiency and excel in forecasting tasks with extended sequences. Furthermore, GNNs such as StemGNN [4], offer competitive results in multivariate time series forecasting by exploring the spectral domain of the data.

Several deep neural network (DNN) models have been proposed for multivariate forecast distributions [6, 24, 25, 31]. A low-rank Gaussian copula model was proposed in [25] using a multi-task univariate LSTM. In [31], a deep factor generative model using a linear combination of RNN latent global factors plus parametric noise was introduced. Normalizing flows for probabilistic forecasting with a multivariate RNN as well as a normalizing flow approach was used in [24]. VRNN was proposed in [6] as a model that uses a variational AE (VAE) in every hidden state of a RNN across the input series. However, such methods suffer from one of the following shortcomings: limited flexibility in modeling distributions in high-dimensional settings [25], only linear combinations of global series and noise distributions are modeled [31], invertible flow needs equal latent and input dimensions [24], and multistep prediction propagation through the whole model is required [6]. Additionally, scaling these methods for high-dimensional multivariate series presents a significant challenge.

3 Problem Setup

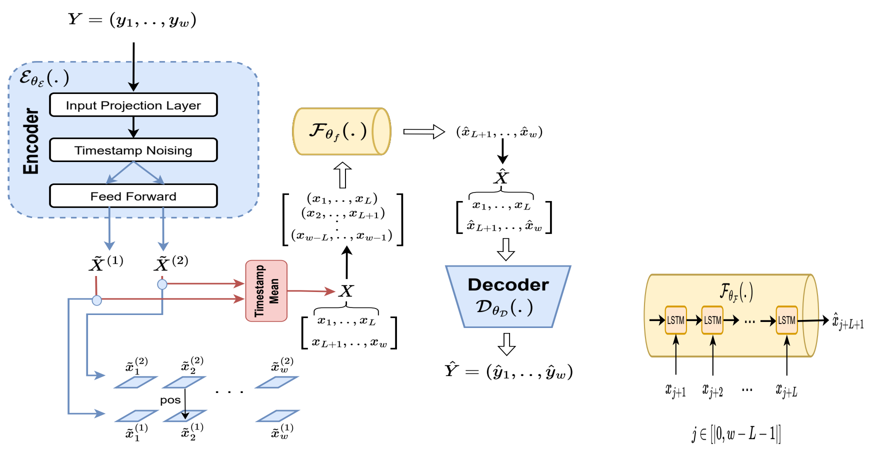
Consider a high-dimensional multivariate time series dataset $\mathbb{Y} \in \mathbb{R}^{T \times N}$, represented by $\mathbb{Y}_{1:T} = (y_1, y_2, \ldots, y_T)^T$, where each time point y_t is a vector of dimensionality N (i.e., $y_t \in \mathbb{R}^N$). The objective is to forecast the next τ values $\mathbb{Y}_{T+1:T+\tau} = (y_{T+1}, y_{T+2}, \ldots, y_{T+\tau})^T$ based on the original time series within the training time-range $Y_{1:T}$. The challenging yet intriguing task is to develop a model capable of capturing the conditional probability distribution in the high-dimensional space:

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

4 Model Architecture

Following TLAE [22], we introduce an autoencoder-like structure for extracting latent series with temporal regularization in the latent space. The complete FR^{3}LS architecture is illustrated in Figure 1. Starting with the input $Y \in \mathbb{R}^{w \times N}$, our model is trained to extract meaningful latent series X. A forecasting model is then employed to predict the next values in the latent series. Finally, a decoder is applied to the latent forecasts, producing forecasts in the original space.

The encoder $\mathcal{E}_{\theta_\mathcal{E}}(.)$ comprises three components: an Input Projection Layer (IPL), a Timestamp Noising (TN) module, and a Feed Forward neural network (FF), inspired by the work in [33]. The Input Projection Layer consists of a fully connected layer that maps each vector $y_t \in \mathbb{R}^N$ at a timestamp to an intermediate latent vector $z_t \in \mathbb{R}^{d_z}$, with $d_z \in \mathbb{N}^*$. The Timestamp Noising module introduces small noise to selected entries of $Z = (z_1, \ldots, z_w)^T$ at randomly chosen timestamps, generating distorted outputs $\tilde{Z}^{(1)}$ and $\tilde{Z}^{(2)}$. These distortions are applied to Z (intermediary subsequence) instead of directly on raw values Y or latent ones X for enhanced model learning stability. The Feed Forward neural network then projects the intermediate latent vectors $\tilde{Z}^{(j)}, j \in \{1, 2\}$ into the two augmented views $\tilde{X}^{(1)}$ and $\tilde{X}^{(2)} \in \mathbb{R}^{w \times d}$. Subsequently, we apply the Timestamp Mean module to produce the latent series given as input to the forecasting model: $X = (x_1, x_2, \ldots, x_w)^T \in \mathbb{R}^{w \times d}$, where $x_t = \frac{\tilde{x}_t^{(1)} + \tilde{x}_t^{(2)}}{2}$ and $d << N$ is the dimensionality of the latent space. It is worth noting that we opted for two context views for model simplification and computational ease, but users can choose a different number if desired


$Y = (y_1, .., y_w)$
$\mathcal{E}_{\theta_\mathcal{E}}(.)$
Encoder
Input Projection Layer
Timestamp Noising
Feed Forward
$\tilde{X}^{(1)}$
$\tilde{X}^{(2)}$
Timestamp Mean
X
$\mathcal{F}_{\theta_f}(.)$
$(\hat{x}_{L+1}, .., \hat{x}_w)$
$\hat{X}$
Decoder
$\mathcal{D}_{\theta_\mathcal{D}}(.)$
$\hat{Y} = (\hat{y}_1, .., \hat{y}_w)$
pos
LSTM
$\hat{x}_{j+L+1}$
x_{j+1} x_{j+2} ... x_{j+L}
$j \in [\![0, w-L-1]\!]$

(a) Global architecture (b) LSTM model

Fig. 1: Model architecture overview.

4.1 Temporal Contextual Consistency

Constructing positive pairs is essential in non-contrastive learning. Following the recommendations of [33], we adopt the temporal contextual consistency paradigm, treating representations at the same timestamp in two augmented contexts as positive pairs to avoid the generation of false positives. We create a context by applying timestamp (light) noising to the intermediary subsequence Z. This approach leverages the fact that timestamp light noising does not alter the magnitude of the time series. Moreover, it encourages the model to learn robust representations at different timestamps capable of reconstructing themselves in distinct contexts.

Timestamp Noising generates an augmented context view for an input series by randomly introducing noise to some of its timestamps. Specifically, it adds small noise to the latent vectors $z_t \in \mathbb{R}^{d_z}$ obtained immediately after the application of the Input Projection Layer, defined as $\tilde{z}_t = z_t + b_t\epsilon_t$ along the time axis. Here, $b_t \in \{0, 1\}$ is a random variable drawn from a Bernoulli distribution with a probability of $p = 0.5$ (i.e., $b_t \sim \mathcal{B}(0.5)$), $\epsilon_t \sim \mathcal{N}(0, 1)$, and both random variables b_t and z_t are independently sampled in every forward pass.

4.2 Non-contrastive Representations Learning

As self-supervised learning (SSL) has demonstrated significant advancements in enabling models to acquire meaningful representations across various domains, Ts2Vec [33] incorporated the concept of contrastive learning to learn time series representations. However, this learning approach is susceptible to selecting false negatives in series with homogeneous distributions, where $p(y_t) \simeq p(y_{t+l})$ for some small integer l. To address this issue, we propose a *non-contrastive* strategy, aiming to encourage the model to learn exclusively from positive samples. This strategy has shown substantial potential in previous works, such as [5, 10, 34].

As depicted in Figure 1a, after generating the augmented views $\tilde{X}^{(1)}, \tilde{X}^{(2)}$ by applying the timestamp noising module followed by the feed forward module, we treat representations at the same timestamp from the two views as positive samples. The *Barlow Twins* loss ($\mathcal{L}_{\mathcal{BT}}$) [34] serves as the loss function describing the non-contrastive error of the model, defined by:

$$\mathcal{L}_{\mathcal{BT}} \triangleq \sum_{i=1}^{d}(1-\mathcal{C}_{ii})^2 + \lambda_{NC}\sum_{i=1}^{d}\sum_{\substack{j=1\\ i\neq j}}^{d}\mathcal{C}_{ij}^2, \tag{2}$$

where $\lambda_{NC} > 0$ and $\mathcal{C}$ is the cross-correlation matrix computed between the two augmented views along the timestamps (batch) dimension:

$$\mathcal{C}_{ij} \triangleq \frac{\sum_t \tilde{x}_{t,i}^{(1)}\tilde{x}_{t,j}^{(2)}}{\sqrt{\sum_t(\tilde{x}_{t,i}^{(1)})^2}\sqrt{\sum_t(\tilde{x}_{t,j}^{(2)})^2}}. \tag{3}$$

This loss function encourages the cross-correlation matrix between embedded outputs to be as close to the identity matrix as possible. Specifically, we aim to equate the diagonal elements to 1, promoting invariance to distortions applied, and the off-diagonal elements to 0, thereby decorrelating different vector components of the embedding and reducing redundant information in the embeddings.

4.3 Deterministic Forecasting

When the latent representation X effectively captures the information in Y, tasks such as forecasting in the original space can be efficiently performed within the much smaller latent space. To this end, we introduce a layer between the encoder and decoder to extract the temporal structure of the latent representations while enforcing forecasting abilities. The central idea is illustrated in Figure 1b: a forecasting model $\mathcal{F}_{\theta_{\mathcal{F}}}(.)$, such as LSTM [11], is employed in the middle layer to capture the long-range dependencies of the embeddings.

The latent matrix $X = (x_1, \ldots, x_w)$ is divided into two subseries: $X_{1:L} = (x_1, \ldots, x_L)$ and $X_{L+1:w} = (x_{L+1}, \ldots, x_w)$. During the training phase, the forecasting model is utilized to estimate the second subsequence $\hat{X}_{L+1:w}$. Subsequences of length $L < w$ denoted by the set $\left\{(x_{j+1}, \ldots, x_{j+L})\middle| j \in [|0, w-L-1|]\right\}$ ($[|a, b|]$ denotes a closed integer interval) serve as inputs to the forecasting model, producing latent forecasts $\hat{X}_{L+1:w} = \left(\hat{x}_{j+L+1} = \mathcal{F}_{\theta_{\mathcal{F}}}((x_{j+1}, ..., x_{j+L}))\right)_{j=0}^{w-L-1}$. The forecasting model is then trained using the deterministic loss $\mathcal{L}\mathcal{F}_{\mathcal{D}}$ with a ℓ_q norm, described as:

$$\mathcal{L}_{\mathcal{F}_{\mathcal{D}}} \triangleq \frac{1}{d(w-L)}\sum_{j=0}^{w-l-1}\|x_{j+L+1} - \mathcal{F}_{\theta_{\mathcal{F}}}\left((x_{j+1}, ..., x_{j+L})\right)\|_{\ell_q}^q. \tag{4}$$

4.4 Probabilistic Forecasting

In high-dimensional settings, probabilistic modeling of forecasts conditioned on observed ones, i.e., $p(y_{T+1},\dots,y_{T+\tau}|y_{1:T})$, poses a significant challenge. Previous research has predominantly focused on either modeling each individual time series independently or considering the joint distribution as Gaussian. However, the former approach neglects inter-series interactions, while the latter suffers from a quadratic increase in the number of learned parameters with the data dimension.

Once again, in the context of probabilistic modeling, we advocate for the non-linear encoding of input data into a significantly lower-dimensional space [22]. Assuming that the encoder function $\mathcal{E}_{\theta_{\mathcal{E}}}$ is sufficiently trained to be considered a one-to-one function, we can then associate the probability of the latent series X with that of the original series Y $(p(x)=p(y))$. Subsequently, we could incorporate a fairly simple probabilistic structure, such as a Gaussian distribution, in the latent space and still be able to model complex distributions of multivariate data through the decoder mapping:

$$p(x_{i+1}|x_{1:i})=\mathcal{N}(x_{i+1};\mu_i,1), \quad i\in[|L,w|]. \tag{5}$$

Here, we identify the conditional distribution as a multivariate Gaussian, with the identity matrix as the covariance matrix to guide the embeddings in capturing different orthogonal patterns in the data. The mean μ_i is computed using the function $\mathcal{F}_{\theta_{\mathcal{F}}}$ as $\mu_i=\mathcal{F}_{\theta_{\mathcal{F}}}(x_1,...,x_i)$. We employ the reparameterization trick [14] to generate latent forecasts needed for backpropagation through input data. In other words, we estimate the value of x_{i+1} using $\hat{x}_{i+1}=\mu_i+1\epsilon=\mathcal{F}_{\theta_{\mathcal{F}}}(x_1,...,x_i)+1\epsilon=\mathcal{R}\circ\mathcal{F}_{\theta_{\mathcal{F}}}(x_1,...,x_i)$, where $\epsilon\sim\mathcal{N}(0,1)$, and $\mathcal{R}(.)$ describes the reparameterization trick function, such that $\mathcal{R}(x)=x+1\epsilon$. Similar to the deterministic setting, the forecasting loss function is defined as $\mathcal{L}_{\mathcal{F}_{\mathcal{P}}}$ using a ℓ_q norm as:

$$\mathcal{L}_{\mathcal{F}_{\mathcal{P}}}\triangleq\frac{1}{d(w-L)}\sum_{j=0}^{w-l-1}\|\hat{x}_{j+L+1}-x_{j+L+1}\|_{\ell_q}^q. \tag{6}$$

4.5 End-to-end training.

After producing elements $\hat{X}_{L+1:w}$, the decoder takes the matrix $\hat{X}=(X_{1:L};\hat{X}_{L+1:w})\in\mathbb{R}^{w\times d}$ as input and generates the matrix $\hat{Y}=(\hat{y}_1,\hat{y}_2,\dots,\hat{y}_w)\in\mathbb{R}^{w\times N}$ as $\hat{Y}=\mathcal{D}_{\theta_{\mathcal{D}}}(\hat{X})$. Consequently, the output $\hat{Y}$ comprises two components: the first consists of elements $\hat{y}_i$, with $i\in[|1,L|]$, decoded directly from the encoder output without passing through the middle layer, defined as $\hat{y}_i=\mathcal{D}_{\theta_{\mathcal{D}}}\circ\mathcal{E}_{\theta_{\mathcal{E}}}(y_i)$, whereas the forecasting model is involved in decoding the second part $\hat{y}_i=\mathcal{D}_{\theta_{\mathcal{D}}}\circ\mathcal{R}^*\circ\mathcal{F}_{\theta_{\mathcal{F}}}\circ\mathcal{E}_{\theta_{\mathcal{E}}}\big((y_{i-L+1},\dots,y_i)\big)$, with $i\in[|L+1,w|]$, and $\mathcal{R}^*(.)\triangleq Identity(.)$ in the point estimate problem or $\mathcal{R}^*(.)\triangleq\mathcal{R}(.)$ otherwise.

Minimizing the error $\mathcal{L}_{\mathcal{AE}}\triangleq\frac{1}{Nw}\|\hat{Y}-Y\|_{\ell_p}^p$ could then be thought of as enabling latent representations to have predictive abilities while also being capable of faithfully reconstructing data. The objective function for a batch Y is defined as:

$$\mathcal{L}_Y(\theta_{\mathcal{E}}, \theta_{\mathcal{F}}, \theta_{\mathcal{D}}) \triangleq \lambda_{AE}\mathcal{L}_{\mathcal{AE}} + \lambda_F \mathcal{L}_{\mathcal{F}} + \lambda_{BT}\mathcal{L}_{\mathcal{BT}}, \tag{7}$$

where $\lambda_{AE}, \lambda_F, \lambda_{BT} \in \mathbb{R}^+$ are positive constants, and $\mathcal{L}_{\mathcal{F}} \triangleq \mathcal{L}_{\mathcal{F}_{\mathcal{D}}}$ for the deterministic case or $\mathcal{L}_{\mathcal{F}} \triangleq \mathcal{L}_{\mathcal{F}_{\mathcal{P}}}$ otherwise.

Once the model is trained, several steps ahead forecasting is performed using rolling windows. Given the past input data $(y_{T-L+1}, \dots, y_T)$, the trained model constructs the latent prediction $\hat{x}_{T+1} = \mathcal{R}^* \circ \mathcal{F}_{\theta_{\mathcal{F}}}\big((x_{T-L+1}, \dots, x_T)\big)$. Subsequently, the predicted point $\hat{y}_{T+1}$ is decoded from $\hat{x}_{T+1}$. This operation can be repeated τ times to predict τ future points of the input time series $\mathbb{Y}$, where we produce the latent prediction $\hat{x}_{T+2}$ by providing the subsequence $(x_{T-L}, \dots, \hat{x}_{T+1})$ as input to the forecasting model.

5 Experiments

Deterministic Forecasting Experimental Setup: For point estimation, we conduct a comparative analysis with state-of-the-art multivariate and univariate forecasting methods, following the approach in [27] and [32]. Our evaluation employs three popular datasets: *electricity* [30]: hourly consumption of 370 houses, *traffic* [7]: hourly traffic on 963 car lanes in San Francisco, and *wiki* [18]: daily web traffic of about 115k Wikipedia articles. We conduct rolling forecasting with 24 time points per window, reserving the last 7 windows for testing in both the traffic and electricity datasets, and 14 points per window with the last 4 windows for testing in the wiki dataset. Evaluation metrics include mean absolute percent error (MAPE), symmetric MAPE (SMAPE), and weighted average percentage error (WAPE) as in [27].

The model architecture and optimization setup align with TLAE, featuring a bottleneck feed-forward network with RELU nonlinearity functions on all layers except the last ones of both the encoder and decoder modules. The dimensions of the layers vary according to the dataset. In the latent space, a 4-layer LSTM network is employed, with 32 hidden units for traffic and wiki datasets, and 64 for electricity, following the recommendations of [22]. The ℓ_1 loss is used in the $\mathcal{L}_{AE}$ loss, and the ℓ_2 loss is used in the $\mathcal{L}_{\mathcal{F}_{\mathcal{D}}}$ loss. Regularization parameters λ_{AE}, λ_F, λ_{BT} are all set to 1, and λ_{NC} is set to 0.005 as suggested in [34]. Additional setup and training details are provided in Tables 1a and 1b as well as sub-section 5.3.

Probabilistic Forecasting Experimental Setup: for the analysis of probabilistic estimation, we introduce two additional datasets: ***solar***: hourly photovoltaic production data from 137 stations used in [15], and ***taxi***: New York taxi rides taken every 30 minutes from 1214 locations [28]. Our evaluation compares the performance of our model against state-of-the-art probabilistic multivariate methods introduced in [22, 25, 31], as well as univariate forecasting methods [16, 23, 26], all utilizing the same data setup. It's essential to note that the data processing and splits utilized in this analysis differ from those of point forecasting. We maintain an identical network architecture as in our previous

Table 1: Statistics and network architectures of datasets.

(a) Statistics of datasets used in both deterministic and probabilistic forecasting experiments.

Dataset	Time Steps T	Dimension N	Predicted Steps τ	Rolling Window k	Frequency
Traffic	10392	963	24	7	hourly
Electricity (large)	25920	370	24	7	hourly
Electricity (small)	5833	370	24	7	hourly
Wiki (large)	635	115084	14	4	daily
Wiki (small)	792	2000	30	5	daily
Solar	7009	137	24	7	hourly
Taxi	1488	1214	24	56	30-min

(b) Network architecture per dataset. Encoder dims = number neurons/layer of the encoder.

Dataset	Encoder dims	LSTM layers	LSTM hidden dim	sequence length L
Traffic	[256, 128, 64]	4	32	32
Electricity (large)	[256, 128, 64]	4	64	32
Electricity (small)	[128, 64]	4	64	32
Wiki (large)	[256, 128, 64]	4	32	16
Wiki (small)	[128, 64]	4	32	64
Solar	[256, 128, 64]	4	32	32
Taxi	[256, 128, 64]	4	32	112

experimental setup and use the same values for regularization parameters, as well as ℓ_2 loss for $\mathcal{L}_{\mathcal{F}_{\mathcal{P}}}$.

To assess the quality of our probabilistic estimates, we employ two distinct error metrics: the first metric is the Continuous Ranked Probability Score across Summed time series (CRPS-Sum) [9,19,22,25], which measures the overall fit of the joint distribution pattern. The second metric is the mean square error (MSE), which assesses the fit of the joint distribution central tendency. Together, these two metrics provide a comprehensive evaluation of the precision of our predictive distribution fit.

5.1 Experimental Results

Table 2a presents a comparison of various deterministic prediction approaches. Results for all models, except the FR^{3}LS model, were originally reported in [22] under the same experimental setup. Here we do not compare our model with classic methods such as VAR, ARIMA etc., as it has already been shown that they obtain performance inferior to TLAE, TRMF and DeepAR methods ([22, 26,32]). Global models leverage global features for multivariate forecasting, while local models employ univariate models to predict individual series separately.

In Table 2b, we display the error scores comparison for probabilistic algorithms. Most results are drawn from Table 2 of [22], with our FR^{3}LS results provided at the end. Conventional statistical multivariate techniques, such as VAR and GARCH ([2,17]), and Vec-LSTM methods, which use a single global LSTM to process and predict all series simultaneously, are included. Additionally, GP methods encompass DNN Gaussian process techniques proposed in [25], with GP-Copula being the primary approach. Further details can be found in [25].

As seen in Table 2a, our method outperforms other global factorization methods in 8 out of 9 dataset-metric combinations. Compared to TLAE, our method achieves an average gain of up to 15% in traffic performance and 13.4% in electricity. Furthermore, compared to other methods, we observe a gain of up to

50% in performance in both the traffic and electricity datasets. For probabilistic forecasting, as shown in Table 2b, our proposed model demonstrates superior performance in the majority of dataset-metric combinations (7 out of 10), with significant gains observed in the Solar, Traffic, and Taxi datasets.

The improvement of the results seen in Tables 2a and 2b over our direct competitor model, TLAE, in both deterministic and probabilistic settings, can be attributed to the enhancement of the latent representations learned by the model. Indeed, enabling the model to be robust against distortions applied to the embeddings enhances its stability in capturing the underlying latent series with predictive power for the time series at hand. Moreover, constraining the model to have embeddings with decorrelated vector components, that is, minimizing the $\mathcal{C}_{ij}^2$ terms in (2), effectively reduces the redundancy of the latent representations. This, in turn, aids the model in focusing on learning latent series that are well-distributed in the latent space.

It is important to note that we utilized LSTM as both the forecasting model and a standard feed-forward autoencoder architecture. We did not employ more advanced models such as TCNs and N-Beats, which could potentially lead to further enhancements. Moreover, in the deterministic prediction case, our model did not use additional local modeling or exogenous features, in contrast to local and combined methods, yet achieved superior performance on 2 out of 3 datasets across all metrics. Finally, we emphasize that our model does not require any further retraining during the testing phase.

5.2 Visualization of latent & original series forecasts.

Figure 2a illustrates the dynamics of trained latent variables and their predictions on the traffic dataset. The blue curve represents the original latent series, while the orange curve depicts their mean predictions. The light-shaded gray area signifies the 90% prediction interval. For each of the 168 predicted timestamps (7×24), 1000 prediction samples were generated. The figure demonstrates that latent variables possess the ability to capture global trends in individual time series. Despite having unique local properties, these latent variables exhibit similar global repeating patterns.

Additionally, Figure 2b showcases a selection of real-time series variables from the traffic dataset alongside their corresponding predictions. The original time series, Y, is depicted in blue, while the predicted time series, $\hat{Y}$, is shown in orange. The light-shaded gray area represents the 90% prediction interval. The predictive power of the latent representations in FR^{3}LS enables the model to accurately capture the overall pattern of the original time series. Furthermore, the model performs well in predicting the local variability associated with individual time series.

5.3 Further Experimental Setup Details

To train the model, we use the Adam optimizer [13] with a learning rate of 0.0001, commonly recommended for stability. Our observations indicate that

Table 2: Comparison of different forecasting algorithms.

(a) Deterministic algorithms comparison in terms of WAPE/MAPE/SMAPE metrics. Best global factorisation results are indicated in bold, best overall performance with *.

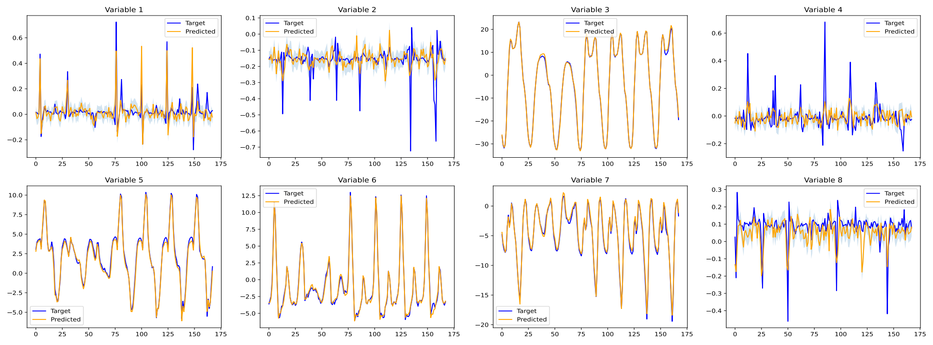
Model	Algorithm	Datasets		
		Traffic	Electricity	Wiki
Global factorisation	**FR^{3}LS** (proposed method)	**0.102***/**0.116***/**0.090***	**0.071***/**0.127***/**0.105***	**0.290**/0.463 /**0.380**
	TLAE [22]	0.117/0.137/0.108	0.080/0.152/0.120	0.334/**0.447**/0.434
	DeepGLO-TCN-MF [27]	0.226/0.284/0.247	0.106/0.525/0.188	0.433/1.59/0.686
	TRMF [32]	0.159/0.226/0.181	0.104/0.280/0.151	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 [27]	0.148/0.168/0.142	0.082/0.341/0.121	0.237/0.441/0.395
	LSTM [11]	0.270/0.357/0.263	0.109/0.264/0.154	0.789/0.686/0.493
	DeepAR [26]	0.140/0.201/0.114	0.086/0.259/0.141	0.429/2.980/0.424
	TCN (no LeveldInit) [1]	0.204/0.284/0.236	0.147/0.476/0.156	0.511/0.884/0.509
	TCN (LeveldInit) [1]	0.157/0.201/0.156	0.092/0.237/0.126	0.212*/0.316*/0.296*
	Prophet [29]	0.303/0.559/0.403	0.197/0.393/0.221	-

(b) Probabilistic comparison in terms of CRPS-Sum/MSE metrics. Lower scores indicate better results. A '-' denotes a method failed (e.g., due to the lack of scalability).

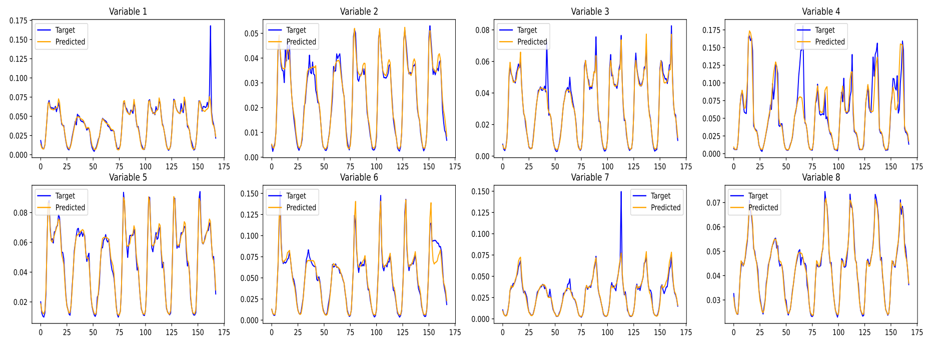
Algorithm	Solar	Electricity-small	Traffic	Taxi	Wiki-small
VAR	0.524 / 7.0e3	0.031 / 1.2e7	0.144 / 5.1e-3	0.292 / -	3.400 / -
GARCH	0.869 / 3.5e3	0.278 / 1.2e6	0.368 / 3.3e-3	- / -	- / -
Vec-LSTM-ind	0.470 / 9.9e2	0.731 / 2.6e7	0.110 / 6.5e-4	0.429 / 5.2e1	0.801 / 5.2e7
Vec-LSTM-ind-scaling	0.391 / 9.3e2	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 [25]	0.828 / 3.7e3	0.947 / 2.7e7	2.198 / 5.1e-1	0.425 / 5.9e1	0.933 / 5.4e7
LSTM-GP-scaling [25]	0.368 / 1.1e3	**0.022** / 1.8e5	0.079 / 5.2e-4	0.183 / 2.7e1	1.483 / 5.5e7
LSTM-GP-Copula [25]	0.337 / 9.8e2	0.024 / 2.4e5	0.078 / 6.9e-4	0.208 / 3.1e1	**0.086** / 4.0e7
VRNN [6]	0.133 / 7.3e2	0.051 / 2.7e5	0.181 / 8.7e-4	0.139 / 3.0e1	0.396 / 4.5e7
TLAE [22]	0.124 / 6.8e2	0.040 / 2.0e5	0.069 / 4.4e-4	0.130 / 2.6e1	0.241 / **3.8e7**
FR^{3}LS (proposed method)	**0.091 / 3.5e2**	0.038 / **1.4e5**	**0.056 / 3.7e-4**	**0.123 / 2.5e1**	0.244 / 3.9e7

higher learning rates often lead to unstable performance. To prevent exploding gradients and stabilize model training, we employ gradient clipping. This technique limits the magnitude of gradients during backpropagation [21]. Furthermore, we implement an adaptive learning rate scheduling strategy to control the training optimization process's convergence while enhancing stability [3].

We adhere to the recommendation of TLAE [22] for setting the subsequence lengths L and w as $w = 2 \times L$. Additionally, when selecting input data for training, a potential approach is to employ sliding windows that overlap entirely, except for one time point, between two subsequences. For instance, we can use two batches, $Y_{t:t+w}$ and $Y_{t+1:t+w+1}$, at times t and $t+1$, respectively, where w represents the input subsequence length. However, to expedite the training process, we choose the nonoverlapping regions as follows: 12, 24, 12, 12, 12, 1, and 12 for the traffic, electricity (large), electricity (small), solar, taxi, wiki (large), and wiki (small) datasets, respectively. In other words, smaller nonoverlapping window sizes were used for smaller datasets.



(a) Latent series forecasting.



(b) Original series forecasting.

Fig. 2: Latent and original series forecasting visualization.

The source code, along with reproducibility instructions for the model and experiments, is publicly available at github.com/Abdallah-Aaraba/FR3LS.

6 Conclusion

This paper introduces an efficient approach for high-dimensional multivariate time series forecasting, advancing the current state-of-the-art in global factorization methods. The method achieves this by combining a flexible nonlinear autoencoder mapping, regularized through a non-contrastive self-supervised learning approach, along with a forecasting model capturing latent temporal dynamics. Furthermore, the proposed approach enables end-to-end training and demonstrates its capability to generate complex predictive distributions by modeling the distribution in the latent space through a nonlinear decoder. Our experiments showcase the superior performance of this method when compared to other state-of-the-art techniques across various commonly used time series datasets. Future research directions may involve exploring alternative temporal models and considering a Transformer-based approach [35] to mitigate the accumulation of forecast errors in sequential predictions and reduce training time.

References

1. Bai, S., Kolter, J.Z., Koltun, V.: An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv preprint arXiv:1803.01271 (2018)
2. Bauwens, L., Laurent, S., Rombouts, J.V.: Multivariate garch models: a survey. Journal of applied econometrics **21**(1), 79–109 (2006)
3. Bottou, L., Curtis, F.E., Nocedal, J.: Optimization methods for large-scale machine learning. SIAM review **60**(2), 223–311 (2018)
4. Cao, D., Wang, Y., Duan, J., Zhang, C., Zhu, X., Huang, C., Tong, Y., Xu, B., Bai, J., Tong, J., et al.: Spectral temporal graph neural network for multivariate time-series forecasting. Advances in neural information processing systems **33**, 17766–17778 (2020)
5. Chen, X., He, K.: Exploring simple siamese representation learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 15750–15758 (2021)
6. Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A.C., Bengio, Y.: A recurrent latent variable model for sequential data. Advances in neural information processing systems **28** (2015)
7. Cuturi, M.: Fast global alignment kernels. In: Proceedings of the 28th international conference on machine learning (ICML-11). pp. 929–936 (2011)
8. Gers, F.A., Schmidhuber, J., Cummins, F.: Learning to forget: Continual prediction with lstm. Neural computation **12**(10), 2451–2471 (2000)
9. Gneiting, T., Raftery, A.E.: Strictly proper scoring rules, prediction, and estimation. Journal of the American statistical Association **102**(477), 359–378 (2007)
10. Grill, J.B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., et al.: Bootstrap your own latent-a new approach to self-supervised learning. Advances in neural information processing systems **33**, 21271–21284 (2020)
11. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
12. Hyndman, R.J., Athanasopoulos, G.: Forecasting: principles and practice. OTexts (2018)
13. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
14. Kingma, D.P., Welling, M.: Stochastic gradient vb and the variational auto-encoder. In: Second international conference on learning representations, ICLR. vol. 19, p. 121 (2014)
15. Lai, G., Chang, W.C., Yang, Y., Liu, H.: Modeling long-and short-term temporal patterns with deep neural networks. In: The 41st international ACM SIGIR conference on research & development in information retrieval. pp. 95–104 (2018)
16. Li, S., Jin, X., Xuan, Y., Zhou, X., Chen, W., Wang, Y.X., Yan, X.: Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. Advances in neural information processing systems **32** (2019)
17. Lütkepohl, H.: New introduction to multiple time series analysis. Springer Science & Business Media (2005)
18. Maggie, Oren, A., Vitaly, K., Will, C.: Web traffic time series forecasting (2017), `https://kaggle.com/competitions/web-traffic-time-series-forecasting`
19. Matheson, J.E., Winkler, R.L.: Scoring rules for continuous probability distributions. Management science **22**(10), 1087–1096 (1976)

20. McKenzie, E.: General exponential smoothing and the equivalent arma process. Journal of Forecasting **3**(3), 333–344 (1984)
21. Mikolov, T., et al.: Statistical language models based on neural networks. Presentation at Google, Mountain View, 2nd April **80**(26) (2012)
22. Nguyen, N., Quanz, B.: Temporal latent auto-encoder: A method for probabilistic multivariate time series forecasting. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 35, pp. 9117–9125 (2021)
23. Rangapuram, S.S., Seeger, M.W., Gasthaus, J., Stella, L., Wang, Y., Januschowski, T.: Deep state space models for time series forecasting. Advances in neural information processing systems **31** (2018)
24. Rasul, K., Sheikh, A.S., Schuster, I., Bergmann, U., Vollgraf, R.: Multivariate probabilistic time series forecasting via conditioned normalizing flows. arXiv preprint arXiv:2002.06103 (2020)
25. Salinas, D., Bohlke-Schneider, M., Callot, L., Medico, R., Gasthaus, J.: High-dimensional multivariate forecasting with low-rank gaussian copula processes. Advances in neural information processing systems **32** (2019)
26. Salinas, D., Flunkert, V., Gasthaus, J., Januschowski, T.: Deepar: Probabilistic forecasting with autoregressive recurrent networks. International Journal of Forecasting **36**(3), 1181–1191 (2020)
27. Sen, R., Yu, H.F., Dhillon, I.S.: Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting. Advances in neural information processing systems **32** (2019)
28. Taxi, N.: New york city taxi and limousine commission (tlc) trip record data. URL: https://www1. nyc. gov/site/tlc/about/tlc-trip-record-data. page (2015)
29. Taylor, S.J., Letham, B.: Forecasting at scale. The American Statistician **72**(1), 37–45 (2018)
30. Trindade, A.: Electricityloaddiagrams20112014 data set. Center for Machine Learning and Intelligent Systems (2015)
31. Wang, Y., Smola, A., Maddix, D., Gasthaus, J., Foster, D., Januschowski, T.: Deep factors for forecasting. In: International conference on machine learning. pp. 6607–6617. PMLR (2019)
32. Yu, H.F., Rao, N., Dhillon, I.S.: Temporal regularized matrix factorization for high-dimensional time series prediction. Advances in neural information processing systems **29** (2016)
33. Yue, Z., Wang, Y., Duan, J., Yang, T., Huang, C., Tong, Y., Xu, B.: Ts2vec: Towards universal representation of time series. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 36, pp. 8980–8987 (2022)
34. Zbontar, J., Jing, L., Misra, I., LeCun, Y., Deny, S.: Barlow twins: Self-supervised learning via redundancy reduction. In: International Conference on Machine Learning. pp. 12310–12320. PMLR (2021)
35. Zerveas, G., Jayaraman, S., Patel, D., Bhamidipaty, A., Eickhoff, C.: A transformer-based framework for multivariate time series representation learning. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. pp. 2114–2124 (2021)
36. Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., Zhang, W.: Informer: Beyond efficient transformer for long sequence time-series forecasting. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 35, pp. 11106–11115 (2021)