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

1st Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address

2nd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address

3rd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address

***Abstract*—Forecasting high-dimensional multivariate time series is an intricate task, both computationally and in terms of accuracy. Prior research has tackled this problem either by treating each variable individually or by neglecting the interactions between the variables. A promising research direction involves methods that leverage scalable matrix factorization for latent-space forecasting, albeit some restricted to linear embeddings and others to limited robustness, focus on the latent dynamics of the whole set of variables. This paper presents a novel methodology exploiting a non-contrastive approach inspired by self-supervised learning methods to guide an autoencoder-like architecture to extract robust latent series while minimizing the redundant information within the embeddings. Such learned representations are used by a temporal forecasting model that produces forecasts within the latent space that are subsequently decoded back to the original space through the decoder. Extensive experiments conducted demonstrate that our model achieves state-of-the-art performance on numerous commonly used multivariate datasets.**

***Index Terms*—time series factorization, probabilistic forecasting, non-contrastive learning**

I. Introduction

Time series forecasting is a crucial task in many industrial real-world scenarios such as retail demand forecasting [1], traffic forecasting [2], and financial prediction [3]. Often in modern datasets, forecasts are needed for many correlated time series, i.e., multivariate time series forecasting, over a long period of time. For instance, electrical consumption forecasting, going from an hour ahead to a week ahead or longer, is of crucial importance for electrical grid operators. Moreover, considering the existing interactions between variables in high-dimensional settings is of great value to the forecasting model, enabling it to capture the overall dynamics of the series.

Most conventional time series forecasting models tackle the problem of predicting time series on an individual basis. For example, industry professionals still commonly use classical statistical forecasting methods such as autoregressive models (AR), ARIMA models [4], exponential smoothing [5], and state-space models (SSMs) [6]. Such approaches have consistently shown better performance in large-scale forecasting competitions compared to machine learning techniques such as RNNs until recently [7]–[9]. Yet, a significant obstacle facing conventional forecasting methods is their limited scalability when dealing with large datasets that may include hundreds of thousands of time series. This is because each individual time series requires its own training.

Deep learning approaches such as Long Short-Tern Memory (LSTM) [10], Gated Recurrent Units (GRU) [11], Temporal Convolution Networks (TCN) [12], try to tackle the shortcomings of individualist conventional methods by means of training on the entire dataset, taking advantage of the shared deep learning model parameters across all series in a multitask univariate forecasting fashion. Capitalizing on this advantage, the winners of the M4 forecast competition [13] used a hybrid ES-RNN [14] where a shared RNN model is used to forecast each series, while seasonal and level exponential smoothing (ES) parameters are simultaneously learned per series to normalize them. However, the deep learning forecasting methods mentioned above possess an inherent limitation in their inability to model inter-series interactions and correlations [15] observed in numerous domains ([16], [17]). For instance, in stock markets, stocks of a particular industry category, such as tech stocks, exhibit shared patterns, and the patterns of different industries can also impact one another.

To address the aforementioned limitations, a promising research direction involves factorizing the relationships between time series into a low-rank matrix, resulting in a summarized latent time series representation [18]–[20]. Temporal Regularized Matrix Factorization (TRMF) [18] proposes factorization of the relationships between time series into a low-rank matrix, thus representing each time series by a linear combination of a small number of latent series. A linear temporal regularization by means of an autoregressive model is imposed on the latent space to ensure temporal dependency between sequential latent

series timestamps. Forecasted values are produced in the latent space and then transformed back through matrix multiplication into the original space. DeepGLO [19] extended this approach by enabling nonlinear regularization of the latent space; This approach consists of training a forecasting model in an iterative fashion, switching between linear matrix factorization training and fitting a latent space TCN. The model's forecasts are then considered as covariates that are subsequently fed, along with other covariates, to another separately trained TCN model.

A recent development in this direction is the Temporal Latent AutoEncoder (TLAE) [20] approach, which extends the previously mentioned methods by incorporating a nonlinear framework through an autoencoder architecture to extract the latent time series space. The forecasts generated by TLAE are also produced within the embedding space before being transformed to the original space through a decoder. Nevertheless, all of the aforementioned factorization methods suffer from robustness issues since they do not account for the possibility of random noise and distortions within the data that could lead to problems like overfitting.

Setting itself apart from prior studies, the TS2Vec model [21] was recently proposed as a framework for learning representations of time series by performing contrastive learning in a hierarchical way over multiple augmented context views to help the model capture robust contextual representations at the timestamp level. This is done through producing two distorted (augmented) views R and R' of an input batch of time series Y by means of a timestamp masking module. The idea of this method is to maximize similarity of representations at the same timestamp of the two augmented views (positive samples) while maximizing dissimilarities between representations at different timestamps (negative samples) in a contrastive learning fashion. While such a method tackles the problem of robustness within latent representations, it is prone to selecting false negatives in series with homogeneous distributions. Thus, taking representations of different timestamps as negative samples is not always appropriate, especially with timestamps close in time. Furthermore, being a representation learning method, TS2Vec is not well aligned with the factorization forecasting line of research, since decoding the latent space back into the original one is not straightforward as in previous methods.

In order to overcome the limitations mentioned above and extend the existing line of research, we propose a forecasting model called FR^{3}LS (a Forecasting model with Robust and Reduced Redundancy Latent Series). As shown in Fig. 1, FR^{3}LS nonlinearly projects high-dimensional time series into a latent space with manageable dimensions, where future value forecasting is performed using latent representations. The output predictions are obtained by decoding latent forecasts generated by the middle layer forecasting model. Such a latent prediction approach helps regularize the embedding space (embedding and latent are used interchangeably in the paper) to capture temporal dependencies between embeddings. Furthermore, latent representations are further regularized using a non-contrastive objective with only positive samples, meaning that the model is trained to produce embeddings that are robust to distortions applied to the input subseries (i.e., augmented context views), while minimizing the redundancy between the components of the vector embedding. Thus, we provide an all-in-one model capable of learning robust representations while maintaining a connection between forecasts in latent and original spaces.

Paper Outline. The remainder of this paper is organized as follows. Section II provides a review of the literature related to the subject. Section III describes the problem statement. Section IV presents our modeling approach. Section V outlines the empirical evaluation setup and our results. Finally, Section VI concludes the paper.

II. Related Work

A. Time Series Forecasting

Traditional time series forecasting methods such as linear SSMs, AR, and ARIMA operate on one time series at a time, which makes them less useful for modern multivariate time series datasets. This paper focuses on recent deep learning approaches beyond traditional methods. For further details on classical methods, we refer the reader to [4]–[6], [22].

Deep learning methods often model each time series individually while learning shared parameters over all series in a dataset. Various models, such as FC-LSTM [15], TCN [12], Transformer-like approaches [23], [24], DeepAR [25], and N-BEATS [26], use techniques like self-attention and residual connections to forecast time series and achieve impressive results in long-horizon forecasting. Transformer and Informer address the efficiency issue of regular self-attention. While effective on certain datasets, these methods have limited use as they cannot capture dependencies between the series.

Several deep neural network (DNN) models have been proposed for multivariate forecast distributions [17], [27]–[29]. [27] proposed a low-rank Gaussian copula model using a multi-task univariate LSTM. [28] proposed a deep factor generative model using a linear combination of RNN latent global factors plus parametric noise. [17] used normalizing flows for probabilistic forecasting with a multivariate RNN as well as a normalizing flow approach. [29] proposed VRNN that uses a variational AE (VAE) in every hidden state of a RNN across the input series. However, such methods suffer from the following shortcomings: [27] has limited flexibility in modeling distributions in high-dimensional settings and is sensitive to rank choice, [28] only models linear combinations of global series and noise distributions and lacks easy local-to-global series mapping, invertible flow in [17] needs equal latent and input dimensions alongside applying the temporal model across all series, and [29] requires multistep prediction propagation through the whole model together with the authors saying that it is intended for high signal-to-noise ratio, while most forecast data are very noisy. Additionally, scaling these methods for high-dimensional multivariate series presents a significant challenge.

B. Self-Supervised Representation of Time Series

Unsupervised representation learning has proven to be a successful approach in a variety of fields including natural language processing [30], [31], computer vision [32]–[34], and speech recognition [35], [36], as evidenced by numerous studies. Several self-supervised time series representation methods have been proposed. SPIRAL [37] constrains learned representations to preserve pairwise similarities in the time domain, while TimeNet [38] trains an encoder-decoder pair to reconstruct the input signal from learned representations. RWS [39] constructs specialized kernels to efficiently generate vector representations of time series. TST [40] utilizes a transformer-based model with a masked MSE loss. However, these methods face challenges in modeling complex and/or long time series.

To address this problem, T-Loss [41] employs negative sampling and a triplet loss to learn scalable representations for multivariate time series. TNC [42], on the other hand, leverages the local smoothness of a signal to define neighborhoods in time while learning generalizable representations. TS-TCC [43] encourages consistency between different data augmentations to improve the quality of learned representations. These methods demonstrate the continued interest and progress in developing self-supervised techniques in time series analysis.

III. Problem Statement

A. Notation

An integer interval $\{1, 2, ..., b\}$ is denoted by $[|1, b|]$. The cardinal of a set S is designated by $|S|$. A multivariate time series dataset is represented by a capital special letter. Let $\mathbb{Y} \in \mathbb{R}^{T \times N}$ be a matrix representing a multivariate time series dataset, with T denoting the number of timestamps and N the number of series. A subseries of $\mathbb{Y}$ is given by the matrix $Y_S = (y_{s_1}, ..., y_{s_w})^T \in \mathbb{R}^{w \times N}$ with $S = \{s_1, ..., s_w \mid 1 \leq s_i \leq T \wedge s_{i+1} = s_i + 1\}$ and $()^T$ meaning the transpose operator. A batch of b subseries of time series $\mathbb{Y}$ is denoted by the set of submatrices $\mathbf{Y}_B = \{Y_{S_1}, ..., Y_{S_b}\}$, with $B = \{S_1, ..., S_b\}$. If $\forall i, j \in [|1, b|] : |S_i| = |S_j|$, then the batch $\mathbf{Y}_B$ is called an equal-sized batch. And the set of possible batches to sample from $\mathbb{Y}$ is denoted $\mathbf{Y}_\mathcal{B}$ (that is, $\mathbf{Y}_B \in \mathbf{Y}_\mathcal{B}$). In the following sections, we also use notation $Y \in \mathbb{R}^{w \times N}$ to simplify Y_S notation as well as to describe a one-batch subseries instead of writing $\mathbf{Y}_B$ when $|B| = 1$. Given two functions f and g, $f \circ g$ designates the composite function $f \circ g(x) = f(g(x))$. The notation $(X_{1:L}; X_{L+1:w})$ is used to express the concatenation of the two subseries $X_{1:L}$ and $X_{L+1:w}$: $(X_{1:L}; X_{L+1:w}) = (x_1, ...x_L, x_{L+1}, ..., x_w)$. Lastly, the symbol $\triangleq$ is used to define a quantity.

B. Problem Setup

Let $\mathbb{Y} \in \mathbb{R}^{T \times N}$ be a high-dimensional multivariate time series dataset denoted by $\mathbb{Y}_{1:T} = (y_1, y_2, ..., y_T)^T$, where each time point y_t is a vector of dimensionality N (i.e., $y_t \in \mathbb{R}^N$). We consider the task of forecasting the next τ values $\mathbb{Y}_{T+1:T+\tau} = (y_{T+1}, y_{T+2}, ..., y_{T+\tau})^T$ given the original time series in the training time-range $Y_{1:T}$. A challenging yet interesting task is to create a model that can model the conditional probability distribution in the high-dimensional space:

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

IV. Model Architecture

Following TLAE [20] model architecture, we propose an autoencoder-like architecture for latent series extraction with temporal regularization in the latent space. Fig. 1 displays the FR^{3}LS architecture in its entirety. To simplify the presentation, we only describe model input as a one batch input subseries $Y \in \mathbb{R}^{w \times N}$ instead of an equally-sized batch input $\mathbf{Y}_B \in \mathbb{R}^{b \times w \times N}$, as the latter case is straightforward. Given the input Y, we train the model to extract meaningful latent series X, a forecasting model is then used to forecast the next values taken by such latent series. Subsequently, a decoder is applied on the forecasted values in order to produce forecasts in the original space.

Following a similar structure to that of Ts2Vec [21], the encoder $\mathcal{E}_{\theta_\mathcal{E}}(.)$ is made up of three components: an Input Projection Layer (IPL), a Timestamp Noising (TN) module, along with a Feed Forward neural network (FF). The Input Projection Layer consists of a fully connected layer that maps the vector $y_t \in \mathbb{R}^N$ at each timestamp to an intermediate latent vector $z_t \in \mathbb{R}^{d_z}$, with $d_z \in \mathbb{N}^*$. The Timestamp Noising module adds small noise to some entries of $Z = (z_1, ..., z_w)^T$ at randomly chosen timestamps, in order to generate augmented context views [21] which yields two outputs $\tilde{Z}^{(1)}$ and $\tilde{Z}^{(2)}$. The Feed Forward neural network is then applied to project intermediate latent vectors $\tilde{Z}^{(j)}, j \in \{1, 2\}$ into two augmented views $\tilde{X}^{(1)}$ and $\tilde{X}^{(2)}$. Subsequently, we apply a Timestamp Mean module in order to produce latent series extracting contextual representations at each timestamp $X = (x_1, x_2, ..., x_w)^T \in \mathbb{R}^{w \times d}$ ($x_t = \frac{\tilde{x}_t^{(1)} + \tilde{x}_t^{(2)}}{2}$), with d being the dimensionality of the latent space.

A. Temporal Contextual Consistency

Selecting a strategy to construct positive pairs is essential in self-supervised learning and more specifically in non-contrastive learning. As argued in [21], previous strategies such as *subseries consistency* [41], *temporal consistency* [42], and *transformation consistency* [43] rely on strong assumptions about the distribution of data and may not be suitable for dealing with time series data, which makes them prone to generating false positive pairs in the presence of level shifts as well as anomalies within the time series. To overcome these issues, we adopt the *temporal contextual consistency* paradigm that treats representations at the same timestamp in two augmented contexts as positive pairs [21]. We generate a context through application of timestamp light noising on the input time series. This way, we benefit from the fact that timestamp light noising does not change the magnitude of the time series, along with encouraging the model to learn

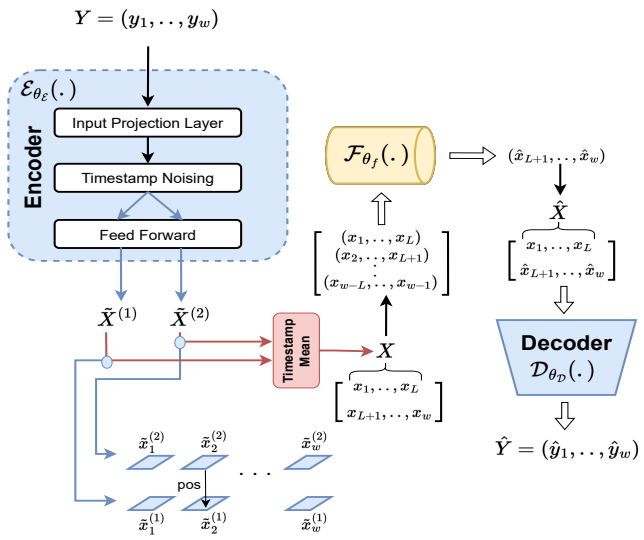

Fig. 1. Model architecture overview

robust representations at different timestamps that are capable of reconstructing themselves in distinct contexts.

Timestamp Noising produces an augmented context view for an input series by randomly noising some of its timestamps. In particular, it adds a small noise to the latent vectors $z_t \in \mathbb{R}^{d_z}$ that we get right after the application of the Input Projection Layer as $\tilde{z}_t = \epsilon_t^{b_t} + z_t$ along the time axis, where b_t is a random variable that follows 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.

B. Non-contrastive Representations Learning

As self-supervised learning (SSL) showed a huge improvement in allowing models to learn meaningful representations recently in different domains such as computer vision, natural language processing, speech recognition, etc., Ts2Vec [21] adopted the concept of contrastive learning in learning time series representations. However, such a learning approach is prone to false negatives selection in series with homogeneous distributions. Thus, we cannot always assume that values taken in different timestamps by a time series are good negative samples. Therefore, to overcome this issue, we propose a *non-contrastive* strategy that encourages the model to learn from positive samples only. Although such an approach might seem counterintuitive, recent non-contrastive SSL methods such as [44]–[46] show its ability to learn good representations regardless of the lack of negative pairs.

As seeded in Fig. 1, after generating the two augmented views $\tilde{X}^{(1)}, \tilde{X}^{(2)} \in \mathbb{R}^{w \times d}$ by means of applying the timestamp masking module followed by the feed forward module, we consider representations at the same timestamp from the two views as positive samples. We use the *Barlow Twins* loss ($\mathcal{L}_{\mathcal{BT}}$) [46] as the loss describing the non-contrastive error of the model as given 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:

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

Such a loss function encourages the cross-correlation matrix between embedded outputs to be as close to the identity matrix as possible. Therefore, we try to equate the diagonal elements to 1, pushing the embeddings to be invariant to the distortions applied on one hand. On the other hand, we try to equate the off-diagonal elements of the cross-correlation matrix to 0, which decorrelates the different vector components of the embedding, thus reducing redundant information in the embeddings.

C. Deterministic Forecasting

The latent representation X that we get after applying the encoder on the input series can be seen as a summarization of the high-dimensional input series Y. Therefore, if X can express reliably Y by means of capturing inter-variables interactions, then tasks such as forecasting in the high-dimensional original space can be performed on the much smaller dimensional latent space. To this end, we add a layer between the encoder and decoder to extract the temporal structure of the latent representations while enforcing them to possess forecasting abilities. The key idea is presented in Fig. 1; A forecasting model such as LSTM [47] is used in the middle layer to capture the long-range dependencies of the embeddings.

As done in [20], the latent matrix $X = (x_1, ..., x_w)$ is decomposed into two subseries: $X_{1:L} = (x_1, ..., x_L)$ and $X_{L+1:w} = (x_{L+1}, ..., x_w)$, and during the training phase we use the forecasting model to estimate the second subsequence $\hat{X}_{L+1:w}$. Subsequences of length $L < w$ denoted by the set $\{(x_{j+1}, ..., x_{j+L}) | j \in [|0, w - L - 1|]\}$ are given as inputs to the forecasting model to produce future points $\hat{X}_{L+1:w} = \left(\hat{x}_{j+L+1} = \mathcal{F}_{\theta_\mathcal{F}}((x_{j+1}, ..., x_{j+L}))\right)_{j=0}^{w-L-1}$. Such a forecasting model is then trained by the deterministic loss $\mathcal{L}_{\mathcal{F}_\mathcal{D}}$ using a l_q norm described as:

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

D. Probabilistic Forecasting

Probabilistically modeling future values conditioned on observed ones, i.e., $p(y_{T+1}, \ldots, y_{T+\tau} | y_{1:T})$, of a high-dimensional time series is a very challenging task. Most prior

research has centered either on the modeling of each individual time series or on the parameterization of the conditional probability of a high-dimensional series through a multivariate Gaussian distribution. Nevertheless, the former kind of approaches does not leverage the inter-series interactions, while the number of learned parameters in the latter ones, specifically the covariance matrix, increases quadratically with the data dimension.

Rather than modeling the data directly in the input space, we propose encoding them into a significantly lower dimensional embedding (as proposed in [20]). This embedding can then serve as the basis for a probabilistic model. If the encoder is sufficiently trained and represents well the original space by capturing nonlinear correlation between variables, we can then associate the probability distribution of the latent series x with that of the original series y, so that we would have $p(x) = p(y)$ by means of assuming that encoder $\mathcal{E}_{\theta_\mathcal{E}}$ function is well trained to be a one-to-one function. Subsequently, we could incorporate fairly simple probabilistic structure, such as a Gaussian distribution, in the latent space and would 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 establish the conditional distribution of latent variables as a multivariate Gaussian, with identity matrix as the covariance matrix to push the embeddings to capture different orthogonal patterns of 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 use the reparameterization trick, as done in variational autoencoder (VAE) models [48], to generate latent forecasts needed to backpropagate through input data. That is, $\hat{x}_{i+1} = \mu_i + 1\epsilon = \mathcal{F}_{\theta_\mathcal{F}}(x_1, ..., x_i) + 1\epsilon = \mathcal{R} \circ \mathcal{F}_{\theta_\mathcal{F}}(x_1, ..., x_i)$, where $\epsilon \sim \mathcal{N}(0, 1)$, and $\mathcal{R}(.)$ describe the reparameterization trick function such that $\mathcal{R}(x) = x + 1\epsilon$.

Similar to the deterministic setting, the forecasting loss function is defined by $\mathcal{L}_{\mathcal{F}_\mathcal{P}}$ using a l_q norm as:

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

with $\hat{x}_{j+L+1} = \mathcal{R} \circ \mathcal{F}_{\theta_\mathcal{F}}\big((x_{j+1}, ..., x_{j+L})\big)$. Furthermore, we must emphasize that, in contrast to VAEs, our model incorporates a temporal model within the latent space and quantifies a conditional discrepancy without imposing a fixed mean constraint.

E. End-to-end training.

After having forecasted elements of $\hat{X}_{L+1:w}$, the decoder will take as input the matrix $\hat{X} = (X_{1:L}; \hat{X}_{L+1:w}) \in \mathbb{R}^{w \times d}$ and produce the matrix $\hat{Y} = (\hat{y}_1, \hat{y}_2, ..., \hat{y}_w) \in \mathbb{R}^{w \times N}$ as $\hat{Y} = \mathcal{D}_{\theta_\mathcal{D}}(\hat{X})$. This way the output $\hat{Y}$ is composed of two components: the first part consists of elements $\hat{y}_i$, with $i \in [|1, L|]$, that are encoded directly from the encoder output without going through the middle layer as a function $\hat{y}_i = \mathcal{D}_{\theta_\mathcal{D}} \circ \mathcal{E}_{\theta_\mathcal{E}}(y_i)$, whereas the forecasting model is involved in the decoding of the second part $\hat{y}_i = \mathcal{D}_{\theta_\mathcal{D}} \circ \mathcal{R}^* \circ \mathcal{F}_{\theta_\mathcal{F}} \circ \mathcal{E}_{\theta_\mathcal{E}}\big((y_{i-L+1}, ..., y_i)\big)$, 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 $\mathcal{L}_{\mathcal{AE}} \triangleq \frac{1}{Nw}\|\hat{Y} - Y\|_{l_p}^p$ could then be thought of as enabling latent representations to have predictive abilities, while at the same time being able to reconstruct data faithfully.

The objective function for one batch Y is defined as:

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

with λ_{AE}, λ_F, and λ_{BT} are constants > 0, 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. The overall loss function is given as

$$\min_{\theta_\mathcal{E}, \theta_\mathcal{F}, \theta_\mathcal{D}} \mathcal{L}(\theta_\mathcal{E}, \theta_\mathcal{F}, \theta_\mathcal{D}) \triangleq \frac{1}{|\mathcal{B}|} \sum_{B \in \mathcal{B}} \mathcal{L}_\mathcal{B}(\theta_\mathcal{E}, \theta_\mathcal{F}, \theta_\mathcal{D}). \tag{8}$$

Minimizing the reconstruction error between Y and $\hat{Y}$ encourages the model to learn cross correlations across individual time series, as well as permits it to produce significant summarization of the high-dimensional original series' space that is encoded in the few latent variables. Whereas, minimizing the divergence between X and $\hat{X}$ nudges the model to grasp the temporal dependency within the latent space with the aim of forecasting future time points. Moreover, minimization of the Barlow Twins loss reinforces the quality and consistency of the timestamp representations.

Once the model is trained, several steps ahead forecasting is done via rolling windows. Given past input data $(y_{T-L+1}, ..., y_T)$, the trained model constructs the latent prediction $\hat{x}_{T+1} = \mathcal{R}^* \circ \mathcal{F}_{\theta_\mathcal{F}}\big((x_{T-L+1}, ..., x_T)\big)$. Subsequently, the predicted point $\hat{y}_{T+1}$ is decoded from $\hat{x}_{T+1}$; The same 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 giving the subsequence $(x_{T-L}, ..., \hat{x}_{T+1})$ as input to the forecasting model.

V. Experiments

A. Deterministic Forecasting Experimental Setup

For point estimation, we compare with state-of-the-art multivariate and univariate forecasting methods as done in [19] and [18]. We use 3 popular datasets: ***electricity*** [49]: describes hourly consumption of 370 houses, ***traffic*** [50]: hourly traffic on 963 car lanes in San Francisco, and ***wiki*** [51]: daily web traffic of about 115k Wikipedia articles. We proceed by performing rolling forecasting with 24 time points per window, last 7 windows are left for testing for both traffic and electricity, and 14 points per window with the last 4 windows for testing for the wiki dataset. Three standard metrics are used to evaluate forecast errors: mean absolute percent error (**MAPE**), symmetric MAPE (**SMAPE**), as well as weighted average percentage error (**WAPE**).

Model architecture and optimization setup are set similarly to TLAE as follows: the autoencoder (encoder and decoder) is a bottleneck feed forward network with RELU nonlinearity functions on all but the last layers of both encoder and decoder modules. The dimensions of the layers vary according to the

dataset. In the latent space, we use a 4-layer LSTM network, with 32 hidden units used for traffic and wiki datasets, and 64 for electricity. The norm used in the $\mathcal{L}_{AE}$ loss is the l_1 loss, and that used in the $\mathcal{L}_{\mathcal{F}_\mathcal{D}}$ is the l_2 loss. Regularization parameters λ_1, λ_2, λ_3 are all set to 1, and λ_{NC} is set to 0.005 as suggested in [46]. Further details on setup and training are provided in Tables I and II, as well as Subsection V-E.

B. Probabilistic Forecasting Experimental Setup

Two additional datasets are added for probabilistic estimation analysis: ***solar***, which comprises hourly photo-voltaic production data from 137 stations used in [52], and ***taxi***, which includes data from New York taxi rides taken every 30 minutes from 1214 locations [53]. We evaluate the performance of our model relative to the state-of-the-art probabilistic multivariate methods introduced in [20], [27], and [28], as well as univariate forecasting methods ([15], [25], [54]), all of which use the same data setup. It is worth noting that the data processing and splits utilized in this analysis differ from those described in Subsection V-A. We employ an identical network architecture as in our previous experimental setup, and use the same values for the regularization parameters, as well as l_2 loss for $\mathcal{L}_{\mathcal{F}_\mathcal{P}}$.

To evaluate 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**) ([27], [55], [56]), 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.

C. Experimental Results

Table III presents the comparison of different deterministic prediction approaches. Except for the results of our proposed FR^{3}LS model, all results were reported in [20] 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 ([18], [20], [25]). Global models exploit global features to provide multivariate forecasting, while local models use univariate models to predict individual series separately.

In Table IV, we illustrate the results of the comparison of the error scores of probabilistic algorithms. Most of the results are taken from Table 2 of [20] with our results at the end (FR^{3}LS). VAR and GARCH are conventional statistical multivariate techniques ([22], [57]), while Vec-LSTM methods employ a single global LSTM that processes and predicts all series simultaneously, utilizing various Gaussian distribution output approaches. Additionally, GP methods include DNN Gaussian process techniques proposed in [27], with GP-Copula being the primary approach. Further details can be found in [27].

As seen in Table III, our method outperforms other global factorization methods in 8/9 dataset-metric combinations. Compared to TLAE, our method can achieve an average gain of up to 15% in traffic performance and 13.4% in electricity. Additionally, compared to the rest of the methods, we can achieve a gain of up to 50% in performance in both the traffic and electricity datasets. Furthermore, for probabilistic forecasting, we can observe in Table IV that our proposed model demonstrates superior performance compared to other methods in the majority of dataset-metric combinations (7/10), with significant gains observed in the Solar, Traffic, and Taxi datasets.

We must mention that we only used LSTM as a forecasting model as well as a standard feed forward autoencoder architecture, and did not utilize more advanced ones such as TCNs and N-Beats, which could potentially lead to further enhancements. Moreover, in the deterministic prediction case, no additional local modeling nor exogenous features were used in the model (in contrast to local and combined methods), and still we were able to achieve superior performance on 2/3 datasets across all metrics. Finally, we emphasize that our model does not require any further retraining during the testing phase.

D. Visualization of latent & original series forecasts.

Fig. 2 illustrates the dynamics of some trained latent variables and their predictions made by the forecasting model on the traffic dataset. The blue curve displays the original latent series, whereas the orange curve illustrates their mean predictions. The light-shaded gray area represents the 90%

TABLE I
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

TABLE II
NETWORK ARCHITECTURE PER DATASET.

Dataset	autoencoder 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

TABLE III
COMPARISON OF DIFFERENT DETERMINISTIC FORECASTING ALGORITHMS IN TERMS OF WAPE/MAPE/SMAPE METRICS. BEST RESULTS FOR GLOBAL MODELING METHODS 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 [20]	0.117/0.137/0.108	0.080/0.152/0.120	0.334/**0.447**/0.434
	DeepGLO-TCN-MF [19]	0.226/0.284/0.247	0.106/0.525/0.188	0.433/1.59/0.686
	TRMF [18]	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 [19]	0.148/0.168/0.142	0.082/0.341/0.121	0.237/0.441/0.395
	LSTM [47]	0.270/0.357/0.263	0.109/0.264/0.154	0.789/0.686/0.493
	DeepAR [25]	0.140/0.201/0.114	0.086/0.259/0.141	0.429/2.980/0.424
	TCN (no LeveldInit) [12]	0.204/0.284/0.236	0.147/0.476/0.156	0.511/0.884/0.509
	TCN (LeveldInit) [12]	0.157/0.201/0.156	0.092/0.237/0.126	0.212*/0.316*/0.296*
	Prophet [58]	0.303/0.559/0.403	0.197/0.393/0.221	-

TABLE IV
COMPARISON OF DIFFERENT PROBABILISTIC FORECASTING ALGORITHMS IN TERMS OF CRPS-SUM/MSE METRICS. LOWER SCORES INDICATE BETTER RESULTS. A '-' DENOTES A METHOD FAILED (E.G., DUE TO EXCESSIVE MEMORY REQUIREMENTS OR LACK OF SCALABILITY FOR THE GIVEN DATA SIZE.)

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 [27]	0.828 / 3.7e3	0.947 / 2.7e7	2.198 / 5.1e-1	0.425 / 5.9e1	0.933 / 5.4e7
LSTM-GP-scaling [27]	0.368 / 1.1e3	**0.022** / 1.8e5	0.079 / 5.2e-4	0.183 / 2.7e1	1.483 / 5.5e7
LSTM-GP-Copula [27]	0.337 / 9.8e2	0.024 / 2.4e5	0.078 / 6.9e-4	0.208 / 3.1e1	**0.086** / 4.0e7
VRNN [29]	0.133 / 7.3e2	0.051 / 2.7e5	0.181 / 8.7e-4	0.139 / 3.0e1	0.396 / 4.5e7
TLAE [20]	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

prediction interval. For each of the 168 predicted timestamps (7 * 24), we generated 1000 prediction samples. The figure indicates that the latent variables have the ability to capture global trends in individual time series. Although they possess unique local properties, these latent variables exhibit similar global repeating patterns.

Furthermore, Fig. 3 illustrates a selection of real time series variables from the traffic dataset alongside their corresponding predictions. The original time series, Y, is indicated in blue, while the predicted time series, $\hat{Y}$, is indicated in orange. The light-shaded gray area depicts 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, as observed. Lastly, the model can predict well the local variability linked with individual time series as well.

E. Further Experimental Setup Details

To train the model, we utilize the Adam [59] optimizer with a commonly used learning rate of 0.0001. Our observations indicate that higher learning rates often lead to unstable performance. The gradient clipping technique is employed to limit the magnitude of gradients during backpropagation to prevent exploding gradients and help stabilize model training [60]. Moreover, we use the learning rate scheduling strategy as well to control the speed and convergence of the training optimization process [61].

The duration of training depends on the dataset size and the number of epochs employed. For example, when working with traffic training data comprising 963 series, each with more than $10,000$ time samples, the training process takes approximately 7 hours to complete with 1000 epochs using 1 NVIDIA Quadro RTX 8000 GPU.

We adhere to the recommendation of TLAE [20] 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,

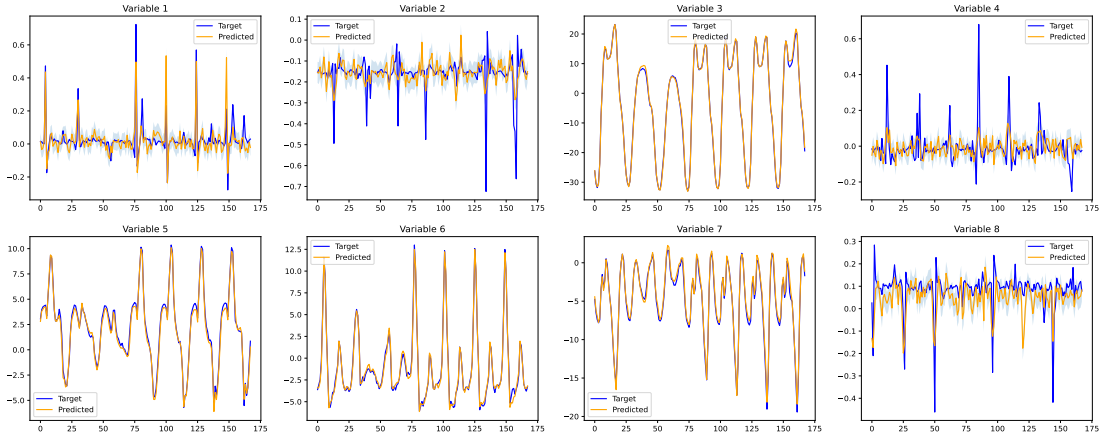

Fig. 2. Latent series forecasting

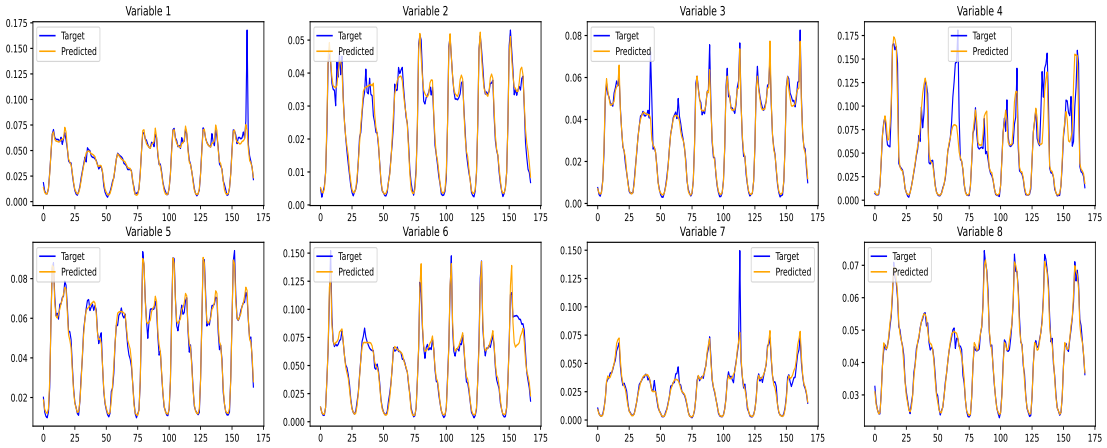

Fig. 3. Original series forecasting

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.

In our experiments, we divide the total dataset $\mathbb{Y} = (y_1, ..., y_{T+k\tau})$ into two sets: the first set, $(y_1, ..., y_T)$, is employed for model training, while the second set, $(y_{T+1}, ..., y_{T+k\tau})$, is reserved for testing. Here, τ is the number of predicted steps (horizon) and k is the number of rolling test windows.

The source code along with reproducibility instructions for the model as well as for the experiments conducted are publicly available at https://github.com/Feshbion/FR3LS

VI. Conclusion

This paper presents an efficient approach for high-dimensional multivariate time series forecasting, thereby elevating the state-of-the-art global factorization approaches. The method achieves this by combining a flexible nonlinear autoencoder mapping, regularized following a noncontrastive self-supervised learning approach, along with an LSTM that models the inherent latent temporal dynamics. On top of that, the proposed approach enables end-to-end training and is capable of generating complex predictive distributions through the nonlinear decoder by modeling the distribution in the latent space. Our experiments demonstrate the superior performance of this method when compared to other state-of-the-art techniques on various commonly used time series datasets. Future research directions may include examining other temporal

models, using a Transformer-based approach [40] to avoid the accumulation of forecast errors in sequential predictions and to decrease training time.

REFERENCES

[1] M. W. Seeger, D. Salinas, and V. Flunkert, "Bayesian intermittent demand forecasting for large inventories," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[2] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," *arXiv preprint arXiv:1707.01926*, 2017.

[3] B. Gui, X. Wei, Q. Shen, J. Qi, and L. Guo, "Financial time series forecasting using support vector machine," in *2014 Tenth International Conference on Computational Intelligence and Security*. IEEE, 2014, pp. 39–43.

[4] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*. OTexts, 2018.

[5] E. McKenzie, "General exponential smoothing and the equivalent arma process," *Journal of Forecasting*, vol. 3, no. 3, pp. 333–344, 1984.

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

[7] S. Makridakis, R. J. Hyndman, and F. Petropoulos, "Forecasting in social settings: The state of the art," *International Journal of Forecasting*, vol. 36, no. 1, pp. 15–28, 2020.

[8] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "Statistical and machine learning forecasting methods: Concerns and ways forward," *PloS one*, vol. 13, no. 3, p. e0194889, 2018.

[9] S. F. Crone, M. Hibon, and K. Nikolopoulos, "Advances in forecasting with neural networks? empirical evidence from the nn3 competition on time series prediction," *International Journal of forecasting*, vol. 27, no. 3, pp. 635–660, 2011.

[10] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," *Neural computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[11] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

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

[13] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "The m4 competition: 100,000 time series and 61 forecasting methods," *International Journal of Forecasting*, vol. 36, no. 1, pp. 54–74, 2020.

[14] S. Smyl, "A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting," *International Journal of Forecasting*, vol. 36, no. 1, pp. 75–85, 2020.

[15] S. S. Rangapuram, M. W. Seeger, J. Gasthaus, L. Stella, Y. Wang, and T. Januschowski, "Deep state space models for time series forecasting," *Advances in neural information processing systems*, vol. 31, 2018.

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

[17] K. Rasul, A.-S. Sheikh, I. Schuster, U. Bergmann, and R. Vollgraf, "Multivariate probabilistic time series forecasting via conditioned normalizing flows," *arXiv preprint arXiv:2002.06103*, 2020.

[18] H.-F. Yu, N. Rao, and I. S. Dhillon, "Temporal regularized matrix factorization for high-dimensional time series prediction," *Advances in neural information processing systems*, vol. 29, 2016.

[19] R. Sen, H.-F. Yu, and I. S. Dhillon, "Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting," *Advances in neural information processing systems*, vol. 32, 2019.

[20] N. Nguyen and B. Quanz, "Temporal latent auto-encoder: A method for probabilistic multivariate time series forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 10, 2021, pp. 9117–9125.

[21] Z. Yue, Y. Wang, J. Duan, T. Yang, C. Huang, Y. Tong, and B. Xu, "Ts2vec: Towards universal representation of time series," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 8, 2022, pp. 8980–8987.

[22] L. Bauwens, S. Laurent, and J. V. Rombouts, "Multivariate garch models: a survey," *Journal of applied econometrics*, vol. 21, no. 1, pp. 79–109, 2006.

[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[24] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, 2021, pp. 11 106–11 115.

[25] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, "Deepar: Probabilistic forecasting with autoregressive recurrent networks," *International Journal of Forecasting*, vol. 36, no. 3, pp. 1181–1191, 2020.

[26] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, "N-beats: Neural basis expansion analysis for interpretable time series forecasting," *arXiv preprint arXiv:1905.10437*, 2019.

[27] D. Salinas, M. Bohlke-Schneider, L. Callot, R. Medico, and J. Gasthaus, "High-dimensional multivariate forecasting with low-rank gaussian copula processes," *Advances in neural information processing systems*, vol. 32, 2019.

[28] Y. Wang, A. Smola, D. Maddix, J. Gasthaus, D. Foster, and T. Januschowski, "Deep factors for forecasting," in *International conference on machine learning*. PMLR, 2019, pp. 6607–6617.

[29] J. Chung, K. Kastner, L. Dinh, K. Goel, A. C. Courville, and Y. Bengio, "A recurrent latent variable model for sequential data," *Advances in neural information processing systems*, vol. 28, 2015.

[30] T. Gao, X. Yao, and D. Chen, "Simcse: Simple contrastive learning of sentence embeddings," *arXiv preprint arXiv:2104.08821*, 2021.

[31] L. Logeswaran and H. Lee, "An efficient framework for learning sentence representations," *arXiv preprint arXiv:1803.02893*, 2018.

[32] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.

[33] E. Xie, J. Ding, W. Wang, X. Zhan, H. Xu, P. Sun, Z. Li, and P. Luo, "Detco: Unsupervised contrastive learning for object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 8392–8401.

[34] H. Xu, X. Zhang, H. Li, L. Xie, W. Dai, H. Xiong, and Q. Tian, "Seed the views: Hierarchical semantic alignment for contrastive representation learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[35] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *Advances in neural information processing systems*, vol. 33, pp. 12 449–12 460, 2020.

[36] Q. Xu, A. Baevski, T. Likhomanenko, P. Tomasello, A. Conneau, R. Collobert, G. Synnaeve, and M. Auli, "Self-training and pre-training are complementary for speech recognition," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 3030–3034.

[37] Q. Lei, J. Yi, R. Vaculin, L. Wu, and I. S. Dhillon, "Similarity preserving representation learning for time series clustering," *arXiv preprint arXiv:1702.03584*, 2017.

[38] P. Malhotra, V. TV, L. Vig, P. Agarwal, and G. Shroff, "Timenet: Pretrained deep recurrent neural network for time series classification," *arXiv preprint arXiv:1706.08838*, 2017.

[39] L. Wu, I. E.-H. Yen, J. Yi, F. Xu, Q. Lei, and M. Witbrock, "Random warping series: A random features method for time-series embedding," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2018, pp. 793–802.

[40] G. Zerveas, S. Jayaraman, D. Patel, A. Bhamidipaty, and C. Eickhoff, "A transformer-based framework for multivariate time series representation learning," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 2114–2124.

[41] J.-Y. Franceschi, A. Dieuleveut, and M. Jaggi, "Unsupervised scalable representation learning for multivariate time series," *Advances in neural information processing systems*, vol. 32, 2019.

[42] S. Tonekaboni, D. Eytan, and A. Goldenberg, "Unsupervised representation learning for time series with temporal neighborhood coding," *arXiv preprint arXiv:2106.00750*, 2021.

[43] E. Eldele, M. Ragab, Z. Chen, M. Wu, C. K. Kwoh, X. Li, and C. Guan, "Time-series representation learning via temporal and contextual contrasting," *arXiv preprint arXiv:2106.14112*, 2021.

[44] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar *et al.*, "Bootstrap your own latent-a new approach to self-supervised learning,"

Advances in neural information processing systems, vol. 33, pp. 21 271–21 284, 2020.

[45] X. Chen and K. He, "Exploring simple siamese representation learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 15 750–15 758.

[46] J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny, "Barlow twins: Self-supervised learning via redundancy reduction," in *International Conference on Machine Learning*. PMLR, 2021, pp. 12 310–12 320.

[47] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[48] D. P. Kingma and M. Welling, "Stochastic gradient vb and the variational auto-encoder," in *Second international conference on learning representations, ICLR*, vol. 19, 2014, p. 121.

[49] A. Trindade, "Electricityloaddiagrams20112014 data set," *Center for Machine Learning and Intelligent Systems*, 2015.

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

[51] "Kaggle wikipedia web traffic," https://www.kaggle.com/c/web-traffic-time-series-forecasting/data, accessed: 2022-11-01.

[52] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling long-and short-term temporal patterns with deep neural networks," in *The 41st international ACM SIGIR conference on research & development in information retrieval*, 2018, pp. 95–104.

[53] N. Taxi, "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.

[54] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang, and X. Yan, "Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting," *Advances in neural information processing systems*, vol. 32, 2019.

[55] J. E. Matheson and R. L. Winkler, "Scoring rules for continuous probability distributions," *Management science*, vol. 22, no. 10, pp. 1087–1096, 1976.

[56] T. Gneiting and A. E. Raftery, "Strictly proper scoring rules, prediction, and estimation," *Journal of the American statistical Association*, vol. 102, no. 477, pp. 359–378, 2007.

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

[58] S. J. Taylor and B. Letham, "Forecasting at scale," *The American Statistician*, vol. 72, no. 1, pp. 37–45, 2018.

[59] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[60] T. Mikolov *et al.*, "Statistical language models based on neural networks," *Presentation at Google, Mountain View, 2nd April*, vol. 80, no. 26, 2012.

[61] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM review*, vol. 60, no. 2, pp. 223–311, 2018.