DeepVARwT: Deep Learning for a VAR Model with Trend

[Link to publication record in Manchester Research Explorer](#)

DeepVARwT: Deep Learning for a VAR Model with Trend

Xixi Li[a], Jingsong Yuan[a,*]

[a]Department of Mathematics, University of Manchester, UK.

Abstract

The vector autoregressive (VAR) model has been used to describe the dependence within and across multiple time series. This is a model for stationary time series which can be extended to allow the presence of a deterministic trend in each series. Detrending the data either parametrically or nonparametrically before fitting the VAR model gives rise to more errors in the latter part. In this study, we propose a new approach called DeepVARwT that employs deep learning methodology for maximum likelihood estimation of the trend and the dependence structure at the same time. A Long Short-Term Memory (LSTM) network is used for this purpose. To ensure the stability of the model, we enforce the causality condition on the autoregressive coefficients using the transformation of Ansley & Kohn (1986). We provide a simulation study and an application to real data. In the simulation study, we use realistic trend functions generated from real data and compare the estimates with true function/parameter values. In the real data application, we compare the prediction performance of this model with state-of-the-art models in the literature.

Keywords: Dependence modeling, VAR, causality condition, trend, deep learning

---

*Corresponding author
Email addresses: `xixi.li@manchester.ac.uk` (Xixi Li), `jingsong.yuan@manchester.ac.uk` (Jingsong Yuan)

1. Introduction

The vector autoregressive (VAR) model has been widely used for multivariate time series modeling. It has desirable properties such as Markovian dependence, and the best linear predictor is particularly simple. For a vector valued time series $\{\mathbf{y}_t\}$ with mean vector $\mathbf{0}$, the VAR(p) model is given by

$$\mathbf{y}_t = A_1\mathbf{y}_{t-1} + A_2\mathbf{y}_{t-2} + \cdots + A_p\mathbf{y}_{t-p} + \varepsilon_t, \qquad t = 0, \pm 1, \pm 2, \ldots, \tag{1}$$

where $A_1, \ldots, A_p$ are constant coefficient matrices, and $\{\varepsilon_t\}$ is a multivariate white noise with mean vector $\mathbf{0}$ and variance-covariance matrix Σ.

The VAR model (1) is for stationary time series. In practice, many time series exhibit nonstationary characteristics. For example, Figure 1 shows three quarterly US macroeconomic series, namely GDP gap, inflation, and federal funds rate, as analyzed by Jordà (2005). Each series is nonstationary as the mean is apparently not constant. A simple approach to detrending a time
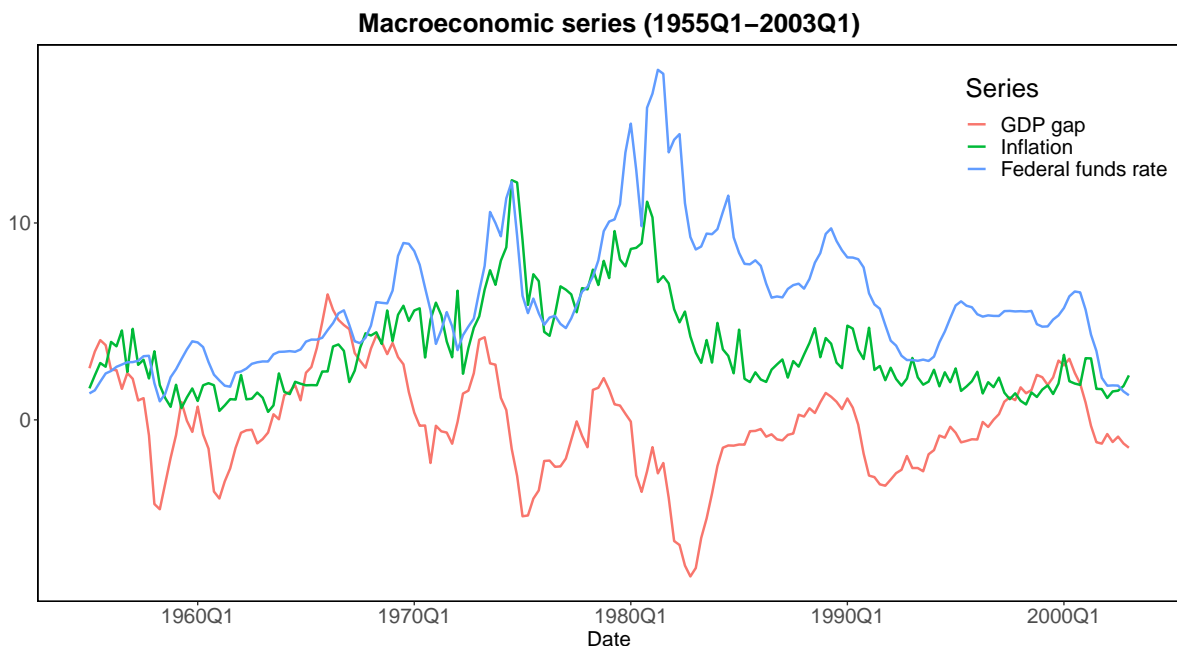
Figure 1. US macroeconomic series spanning 1955Q1 to 2003Q1.

series is to difference it until it appears to be stationary. This is effective when the trend is a low order polynomial. However, the trend itself may be of interest, and modeling it together with the dependence structure can be preferable. The former can be estimated by smoothing the data, using methods such as Kernel Smoothing (Wand & Jones, 1994), Locally Weighted Scatterplot Smoothing (Lowess), or Smoothing Splines, to name just a few. The series after removing the trend in each component can then be analyzed by fitting a VAR model. Inference on model parameters will have to ignore errors in estimating the trend in this semi-parametric approach.

The VAR model can be extended to include exogenous variables $\mathbf{x}_t$ that are deterministic functions of time t. For example, $\mathbf{x}_t = (1, t, t^2)'$, and model (1) becomes

$$\mathbf{y}_t = A_1\mathbf{y}_{t-1} + A_2\mathbf{y}_{t-2} + \cdots + A_p\mathbf{y}_{t-p} + C\mathbf{x}_t + \varepsilon_t, \qquad t = 0, \pm 1, \pm 2, \ldots, \tag{2}$$

where C is a matrix. This is equivalent to allowing the mean μ_t of $\mathbf{y}_t$ to be quadratic in t while $\{\mathbf{y}_t - \mu_t\}$ satisfies the VAR model (1). Both the trend and dependence parameters are estimated simultaneously by ordinary least squares (Pfaff & Stigler, 2018).

In this paper, we model the mean μ_t by a recurrent neural network of the LSTM (Long Short-Term Memory) type, with input x_t at time t, and simultaneously $\{\mathbf{y}_t - \mu_t\}$ by the VAR model (1). Details are given in (3), (5) and (6). This way, the deterministic trends can be highly nonlinear and the Markovian dependence in $\{\mathbf{y}_t\}$ is retained. All the parameters will come from the network, which will be trained in such a way that the exact Gaussian log-likelihood is maximised, so that inference can be made using the fitted model. In particular, future values of $\mathbf{y}_t$ can be predicted and prediction intervals constructed.

The rest of the paper is organized as follows: Section 2 reviews the literature on VAR modeling and recent deep learning based statistical forecasting methods. Section 3 describes trend generation using an LSTM network, VAR parameterization, the Gaussian log-likelihood function, and its use in network training. Section 4 is a simulation study using trends generated from real data. Section 5 shows results of model fitting to a real data set and comparisons with alternative models in terms of forecasting accuracy. Section 6 offers concluding remarks.

2. Literature review

In this section, we review some related literature on VAR modeling and deep learning based statistical forecasting.

2.1. VAR modeling and causality

It is usually assumed that model (1) is causal in the sense that $\mathbf{y}_t$ can be expressed linearly in terms of ε_t, $\varepsilon_{t-1}, \ldots$, so that ε_t is the innovation or one-step-ahead prediction error corresponding to the best linear predictor $\hat{\mathbf{y}}_t = A_1\mathbf{y}_{t-1} + A_2\mathbf{y}_{t-2} + \cdots + A_p\mathbf{y}_{t-p}$ of $\mathbf{y}_t$ in terms of $\mathbf{y}_{t-1}, \mathbf{y}_{t-2}, \ldots$. The causality condition is that all the roots of $\det(I - A_1 z - A_2 z^2 - \cdots - A_p z^p)$ lie outside the unit circle (Hannan, 1970). It also ensures that the linear system given by (1) is stable in the sense that bounded input leads to bounded output.

The parameter space of a causal VAR model is highly complicated. In the univariate case it can be mapped to (0,1) in each dimension using partial correlations, see Barndorff-Nielsen & Schou

(1973). Work on the multivariate case include Morf et al. (1978), Ansley & Kohn (1986), Roy et al. (2019) and Heaps (2022). We follow the procedures of Ansley & Kohn (1986) and give the details in Section 3.4.

2.2. Recent deep learning based statistical forecasting

Wang et al. (2019) assumed a global (non-random) factor given by a recurrent neural network and a random effects component with a Gaussian structure. Salinas et al. (2020) proposed a DeepAR model to train a recurrent network on a large number of related time series $\{y_t^{(i)}\}$, with the most recent value $y_{t-1}^{(i)}$ and covariates $\mathbf{x}_t^{(i)}$ as input. The hidden state $\mathbf{h}_t^{(i)}$ is used to produce the (conditional) mean $\mu_t^{(i)}$ and standard deviation $\sigma_t^{(i)}$ of $y_t^{(i)}$. The same network parameters are shared between the individual series. Rangapuram et al. (2018) proposed a DeepState model to learn a shared mapping from the covariates $\mathbf{x}_t^{(i)}$ to the parameters of a state space model.

All these methods require the time series to be independent so that the loss function can be written in a simple additive form, thus leaving out dependence information across the series. In further developments, Salinas et al. (2019) combined an RNN-based time series model with a Gaussian copula process. de Bézenac et al. (2020) proposed a Normalizing Kalman Filter (NKF) model, alleviating assumptions like Gaussianity and linear dependence by augmenting state space models with normalizing flows (Rezende & Mohamed, 2015) to capture dependence between the time series.

3. DeepVARwT: Deep learning for VAR(p) with trend

In this study, we consider a VAR(p) model with trend for a vector time series $\{\mathbf{y}_t\}$, which is written as

$$\mathbf{y}_t - \boldsymbol{\mu}_t = A_1(\mathbf{y}_{t-1} - \boldsymbol{\mu}_{t-1}) + A_2(\mathbf{y}_{t-2} - \boldsymbol{\mu}_{t-2}) + \cdots + A_p(\mathbf{y}_{t-p} - \boldsymbol{\mu}_{t-p}) + \boldsymbol{\varepsilon}_t, \qquad t = 0, \pm 1, \pm 2 \ldots . \tag{3}$$

Here $\mathbf{y}_t = (y_{1,t}, \ldots, y_{m,t})'$ is an $m \times 1$ random vector, $\boldsymbol{\mu}_t = (\mu_{1,t}, \ldots, \mu_{m,t})'$ is an $m \times 1$ mean vector, $A_1, \ldots, A_p$ are $m \times m$ coefficient matrices, and $\{\boldsymbol{\varepsilon}_t\}$ is an i.i.d. Gaussian vector white noise with mean vector $\mathbf{0}$ and variance-covariance matrix Σ. It is also assumed that as a result of causality, $\boldsymbol{\varepsilon}_t$ is uncorrelated with $\mathbf{y}_{t-1}, \mathbf{y}_{t-2}, \ldots$, so that the RHS of (3) consists of the best linear predictor $\hat{\mathbf{y}}_t - \boldsymbol{\mu}_t$ of $\mathbf{y}_t - \boldsymbol{\mu}_t$ in terms of $\mathbf{y}_{t-1} - \boldsymbol{\mu}_{t-1}, \mathbf{y}_{t-2} - \boldsymbol{\mu}_{t-2}, \ldots$, and the prediction error $\boldsymbol{\varepsilon}_t$.

The trend $\boldsymbol{\mu}_t$ as well as $A_1, \ldots, A_p$ and Σ will all come from an LSTM network which is described in the following.

3.1. Long Short-Term Memory (LSTM)

A recurrent neural network (RNN) is designed in such a way that the current output is determined by the previous output and the current input (Sak et al., 2014). However, standard RNNs are limited in the range of contextual information (Gers et al., 2000). There exists a vanishing gradient problem in standard RNNs (Graves et al., 2008). To handle these issues, a special RNN structure called Long Short-Term Memory (LSTM) uses memory cells and gate units to guarantee constant error flow within its constant error carrousel (Hochreiter & Schmidhuber, 1997; Gers et al., 2000).

At time t, the input gate $\mathbf{i}_t$, forget gate $\mathbf{f}_t$ and output gate $\mathbf{o}_t$ are determined by the current input $\mathbf{x}_t$ and the last hidden state $\mathbf{h}_{t-1}$. The memory cell $\mathbf{c}_t$ puts information from the last memory cell $\mathbf{c}_{t-1}$ through the forget gate $\mathbf{f}_t$ and information from the candidate memory cell $\tilde{\mathbf{c}}_t$ through the input gate $\mathbf{i}_t$. The output gate $\mathbf{o}_t$ decides how much information from the memory cell $\mathbf{c}_t$ should contribute to the hidden state $\mathbf{h}_t$. Figure 2 shows the computation unit for the hidden state $\mathbf{h}_t$ in an LSTM network and the corresponding calculations are as follows (Gers et al., 2000).

$$
\begin{aligned}
\textbf{Input gate:} \quad & \mathbf{i}_t = \sigma\left(W_{xi}\mathbf{x}_t + W_{hi}\mathbf{h}_{t-1} + \mathbf{b}_i\right), \\
\textbf{Forget gate:} \quad & \mathbf{f}_t = \sigma\left(W_{xf}\mathbf{x}_t + W_{hf}\mathbf{h}_{t-1} + \mathbf{b}_f\right), \\
\textbf{Output gate:} \quad & \mathbf{o}_t = \sigma\left(W_{xo}\mathbf{x}_t + W_{ho}\mathbf{h}_{t-1} + \mathbf{b}_o\right), \\
\textbf{Candidate memory cell:} \quad & \tilde{\mathbf{c}}_t = \tanh\left(W_{xc}\mathbf{x}_t + W_{hc}\mathbf{h}_{t-1} + \mathbf{b}_c\right), \\
\textbf{Memory cell:} \quad & \mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t, \\
\textbf{Hidden state:} \quad & \mathbf{h}_t = \mathbf{o}_t \odot \tanh\left(\mathbf{c}_t\right),
\end{aligned}
\tag{4}
$$

where W_{xi}, W_{xf}, W_{xo}, W_{hi}, W_{hf}, W_{ho}, W_{xc} and W_{hc} are weight parameters, $\mathbf{b}_i$, $\mathbf{b}_f$, $\mathbf{b}_o$ and $\mathbf{b}_c$ are bias parameters, $\sigma()$ is an activation function and the operator $\odot$ denotes the element-wise product.
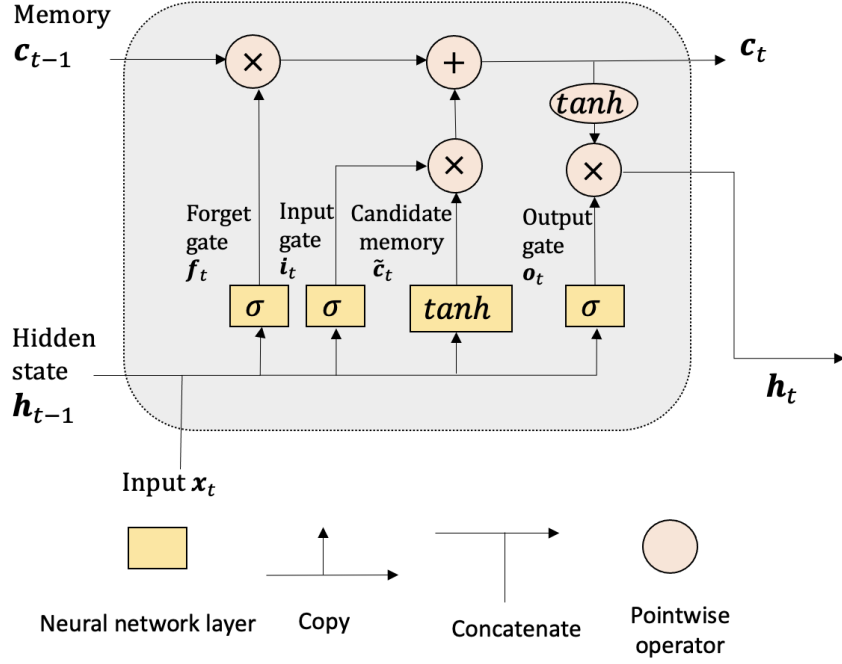
Figure 2. The computation unit for hidden state $\boldsymbol{h}_t$ in an LSTM.

3.2. Time-dependent trend generation using LSTM

We map the hidden state $\mathbf{h}_t$ to the trend term $\boldsymbol{\mu}_t$ via affine transformations as follows. First, calculate

$$\mathbf{h}_t = \text{LSTM}(\mathbf{h}_{t-1}, \mathbf{x}_t; \boldsymbol{\Phi}), \tag{5}$$

according to (4), then

$$\boldsymbol{\mu}_t = W_\mu \mathbf{h}_t + \mathbf{b}_\mu, \tag{6}$$

where $\mathbf{x}_t = (t, t^2, t^3)'$ (say, suitably scaled), $\boldsymbol{\Phi}$ contains the weight and bias parameters of the LSTM network, W_μ is a matrix of weights and $\mathbf{b}_\mu$ is a vector of bias parameters.

3.3. VAR parameter generation

We allocate $m^2 p$ parameters in the neural network to form the coefficient matrices $A_1, ..., A_p$, and another $m(m+1)/2$ parameters to form a lower triangular matrix L. The latter is used to construct $\Sigma = LL'$ for the innovation variance-covariance matrix in (3). These parameters are initialized and updated by the network together with other network parameters. The answers for $A_1, ..., A_p$ and L then go through the next step of reparameterization.

3.4. Reparameterizing VAR(p) to enforce causality

Given a set of VAR coefficient matrices $A_1, ..., A_p$, we transform them using the algorithm of Ansley & Kohn (1986) in the following two steps, so that the causality condition is satisfied.

6

- **Partial autocorrelation matrix construction.** For $j = 1, \ldots, p$, find the Cholesky factorisation $I + A_j A_j' = B_j B_j'$, then compute

$$P_j = B_j^{-1} A_j \qquad (7)$$

as partial autocorrelation matrices (Ansley & Kohn, 1986).

- **Causal VAR coefficient generation.** The Ansley & Kohn (1986) algorithm maps the partial autocorrelation matrices $\{P_j\}$ into the coefficient matrices $\{A_{s,i}\}$ and $\{A_{s,i}^*\}$ for forward and backward predictions using s past/future values, with prediction error variance-covariance matrices Σ_s and Σ_s^* respectively.

Initialization: Make $\Sigma_0 = \Sigma_0^* = I$, and $L_0 = L_0^* = I$, I being the identity matrix.

Recursion: For $s = 0, \ldots, p-1$,

- Compute

$$A_{s+1,s+1} = L_s P_{s+1} (L_s^*)^{-1}, \quad A_{s+1,s+1}^* = L_s^* P_{s+1}' L_s^{-1}. \qquad (8)$$

- For $i = 1, \ldots, s$ ($s > 0$), compute

$$
\begin{aligned}
A_{s+1,i} &= A_{s,i} - A_{s+1,s+1} A_{s,s-i+1}^*, \\
A_{s+1,i}^* &= A_{s,i}^* - A_{s+1,s+1}^* A_{s,s-i+1}.
\end{aligned}
\qquad (9)
$$

- Compute

$$\Sigma_{s+1} = \Sigma_s - A_{s+1,s+1} \Sigma_s^* A_{s+1,s+1}', \qquad (10)$$

$$\Sigma_{s+1}^* = \Sigma_s^* - A_{s+1,s+1}^* \Sigma_s \left(A_{s+1,s+1}^* \right)', \qquad (11)$$

and obtain their Cholesky factorizations $L_{s+1} L_{s+1}'$ and $L_{s+1}^* L_{s+1}^{*\prime}$ respectively.

Causal VAR coefficients: Obtain the Cholesky factorization $L_p L_p'$ of Σ_p, then take

$$A_i = (L L_p^{-1}) A_{p,i} (L L_p^{-1})^{-1}, \quad i = 1, \ldots, p \qquad (12)$$

as new VAR(p) coefficient matrices.

3.5. The Gaussian log-likelihood

Given that the time series $\{\mathbf{y}_t\}$ has a Gaussian structure and AR(p) dependence, the likelihood of $\mathbf{y} = (\mathbf{y}_1', \ldots, \mathbf{y}_T')'$ can be written as

$$L(\mathbf{\Theta}; \mathbf{y}) = f(\mathbf{y}_1, \ldots, \mathbf{y}_p) \prod_{t=p+1}^{T} f(\mathbf{y}_t | \mathbf{y}_{t-1}, \ldots, \mathbf{y}_{t-p}), \qquad (13)$$

using the joint normal density $f(\mathbf{y}_1, ..., \mathbf{y}_p)$ and the conditional Gaussian densities $f(\mathbf{y}_t | \mathbf{y}_{t-1}, ..., \mathbf{y}_{t-p})$, $t = p + 1, ..., T$, where $\boldsymbol{\Theta}$ consists of the parameters $\boldsymbol{\Psi}_1 = \{\boldsymbol{\Phi}, W_\mu, \boldsymbol{b}_\mu\}$ for trend generation and $\boldsymbol{\Psi}_2 = \{A_1, ..., A_p, L\}$ for the VAR coefficient matrices $A_1, ..., A_p$ and the Σ matrix. The log-likelihood is

$$\ell(\boldsymbol{\Theta}; \mathbf{y}) = -\frac{1}{2}\left[n\log(2\pi) + \log|R_p| + \left(\mathbf{y}_{1:p} - \boldsymbol{\mu}_{1:p}\right)' R_p^{-1} \left(\mathbf{y}_{1:p} - \boldsymbol{\mu}_{1:p}\right) + (T - p)\log|\Sigma| + \sum_{t=p+1}^{T} \varepsilon_t' \Sigma^{-1} \varepsilon_t \right], \tag{14}$$

where $n = mT$, R_p is the variance-covariance matrix of $\mathbf{y}_{1:p} = (\mathbf{y}_1', ..., \mathbf{y}_p')'$ obtained using standard results (Lütkepohl, 2005), $\boldsymbol{\mu}_{p+1}, ..., \boldsymbol{\mu}_T$ from the output of the neural network and the VAR coefficient matrices $A_1, ..., A_p$ are used for the calculation of $\varepsilon_{p+1}, ..., \varepsilon_T$ according to (3).

3.6. Network training

We employ the popular tool pytorch (Paszke et al., 2019) for network training, after setting up its structure. The parameters of the neural network are updated by a modified gradient descent (GD) algorithm AdaGrad (Adaptive Gradient, Duchi et al., 2011). The basic idea of gradient descent is to follow the opposite direction of the gradient of the loss function at the current point. Compared with conventional gradient descent algorithms, AdaGrad provides individual adaptive learning rates for different parameters. At iteration k, the learning rate is modified by the diagonal elements of $G = \sum_{\tau=1}^{k} \mathbf{g}^{(\tau)} \mathbf{g}^{(\tau)'}$, where $\mathbf{g}^{(\tau)}$ is the gradient of the loss function at iteration τ.

- **Phase 1.** In order to speed up the training process, we first employ OLS (ordinary least squares) to obtain trend parameters $\boldsymbol{\Psi}_{\text{ols}}$ to be used as initial values in Phase 2.
- **Phase 2.** Then the loss function becomes minus the log-likelihood. The trend parameters are fine-tuned with a smaller learning rate so that the trend terms get updated in small steps to avoid large changes that affect the estimation of the VAR parameters.

Details of the GD based training procedure in Phase 2 are presented in Algorithm 1.

3.7. Prediction from trained network

We continue to run the trained network for $t = T + 1$, $T + 2$ etc to generate future trend values $\boldsymbol{\mu}_t$ and produce point forecasts using the formula (A.9) in the Appendix.

Approximate 95% prediction intervals can be obtained by adding and subtracting 1.96 times the standard deviations of prediction errors using results in the Appendix.

Algorithm 1 GD based network training for VAR with trend

Input:

Time series observations $\mathbf{y}_1,\ldots,\mathbf{y}_T$;

Values of t functions $\mathbf{x}_1,\ldots,\mathbf{x}_T$;

Initial parameters $\boldsymbol{\Psi}_{ols}$;

Initial parameters $\{A_1^{(0)},\ldots,A_p^{(0)},L^{(0)}\}$;

Initial hidden state $\mathbf{h}_0$;

Learning rates η_1 and η_2;

Number of iterations K;

Precision value *prec* for the stopping criteria.

Output:

The optimal network parameters $\boldsymbol{\Theta}^{\star} = (\boldsymbol{\Psi}_1^{\star}, \boldsymbol{\Psi}_2^{\star})$.

1: Set initial values $\boldsymbol{\Psi}_1^{(0)} = \boldsymbol{\Psi}_{ols}$, $\boldsymbol{\Psi}_2^{(0)} = \{A_1^{(0)},\ldots,A_p^{(0)},L^{(0)}\}$, initial hidden state $\mathbf{h}_0 = \mathbf{0}$, learning rates η_1, η_2, and number of iterations K.

2: **while** $k \le K$ and $(rc_1 > prec$ or $rc_2 > prec)$ **do**   ▷ Iterations for MLE optimization.

3:      **for** $t \leftarrow 1$ to T **do**

4:          Compute the hidden state $\mathbf{h}_t^{(k)} = \text{LSTM}\left(\mathbf{h}_{t-1}^{(k)}, \mathbf{x}_t; \boldsymbol{\Phi}^{(k)}\right)$.

5:          Compute the trend term $\boldsymbol{\mu}_t^{(k)} = W_\mu^{(k)}\mathbf{h}_t^{(k)} + \mathbf{b}_\mu^{(k)}$.

6:      **end for**

7:      Compute $P_1^{(k)},\ldots,P_p^{(k)}$ from $A_1^{(k)},\ldots,A_p^{(k)}$ using (7).

8:      Transform $P_1^{(k)},\ldots,P_p^{(k)}$ into new $A_1^{(k)},\ldots,A_p^{(k)}$ using (8) to (12).

9:      Compute $\Sigma^{(k)} = L^{(k)}L^{(k)\prime}$.

10:     Evaluate the loss function $-\ell(\boldsymbol{\Theta}^{(k)};\mathbf{y})$ at $\boldsymbol{\Theta}^{(k)} = (\boldsymbol{\Psi}_1^{(k)}, \boldsymbol{\Psi}_2^{(k)})$ using (14).

11:     Compute relative change of log-likelihood $rc_1 = \left|\dfrac{\ell(\boldsymbol{\Theta}^{(k-1)};\mathbf{y}) - \ell(\boldsymbol{\Theta}^{(k-2)};\mathbf{y})}{\ell(\boldsymbol{\Theta}^{(k-2)};\mathbf{y})}\right|$.

12:     Compute relative change of log-likelihood $rc_2 = \left|\dfrac{\ell(\boldsymbol{\Theta}^{(k)};\mathbf{y}) - \ell(\boldsymbol{\Theta}^{(k-1)};\mathbf{y})}{\ell(\boldsymbol{\Theta}^{(k-1)};\mathbf{y})}\right|$.

13:     Compute gradient of loss function $\mathbf{g}_1^{(k)} = \frac{\partial}{\partial \boldsymbol{\Psi}_1}(-\ell(\boldsymbol{\Theta};\mathbf{y}))\big|_{\boldsymbol{\Theta}=\boldsymbol{\Theta}^{(k)}}$.

14:     Compute gradient of loss function $\mathbf{g}_2^{(k)} = \frac{\partial}{\partial \boldsymbol{\Psi}_2}(-\ell(\boldsymbol{\Theta};\mathbf{y}))\big|_{\boldsymbol{\Theta}=\boldsymbol{\Theta}^{(k)}}$.

15:     Compute $G_1 = \sum_{\tau=1}^{k} \mathbf{g}_1^{(\tau)}\mathbf{g}_1^{(\tau)\prime}$, $G_2 = \sum_{\tau=1}^{k} \mathbf{g}_2^{(\tau)}\mathbf{g}_2^{(\tau)\prime}$.

16:     Update parameters $\boldsymbol{\Psi}_1^{(k+1)} = \boldsymbol{\Psi}_1^{(k)} - \eta_1\text{diag}(G_1)^{-\frac{1}{2}}\odot\mathbf{g}_1^{(k)}$.   ▷ ($\odot$ is an element-wise multiplication operator between two vectors.)

17:     Update parameters $\boldsymbol{\Psi}_2^{(k+1)} = \boldsymbol{\Psi}_2^{(k)} - \eta_2\text{diag}(G_2)^{-\frac{1}{2}}\odot\mathbf{g}_2^{(k)}$.

18:     $k \leftarrow k+1$

19: **end while**

20: The optimal parameters are $\boldsymbol{\Theta}^{\star} = (\boldsymbol{\Psi}_1^{(k)}, \boldsymbol{\Psi}_2^{(k)})$.

---

4. Simulation study

To assess the finite sample performance of the deep learning based maximum likelihood estimation method, we simulated 100 samples each of size $T = 800$ from the DeepVARwT model ($p = 2$, $m = 3$)

$$\mathbf{y}_t - \boldsymbol{\mu}_t = A_1(\mathbf{y}_{t-1} - \boldsymbol{\mu}_{t-1}) + A_2(\mathbf{y}_{t-2} - \boldsymbol{\mu}_{t-2}) + \boldsymbol{\varepsilon}_t, \tag{15}$$

where

$$A_1 = \begin{pmatrix} -1.0842 & -0.1245 & 0.3137 \\ -0.7008 & -0.3754 & -0.2064 \\ 0.3166 & 0.3251 & 0.2135 \end{pmatrix}, \quad A_2 = \begin{pmatrix} -0.5449 & -0.3052 & -0.1952 \\ -0.4057 & 0.5129 & 0.3655 \\ 0.0054 & -0.2911 & 0.2066 \end{pmatrix},$$

and

$$\Sigma = \begin{pmatrix} 0.4834 & -0.2707 & 0.1368 \\ -0.2707 & 0.4079 & -0.0221 \\ 0.1368 & -0.0221 & 0.4103 \end{pmatrix}$$

is the variance-covariance matrix of ε_t. Realistic values for the trend term μ_t are obtained from daily closing prices of three US stocks from 3rd October 2016 to 5th December 2019 via kernel smoothing.

An example of the simulated multiple series is shown in Figure 3, each having clearly a trend.
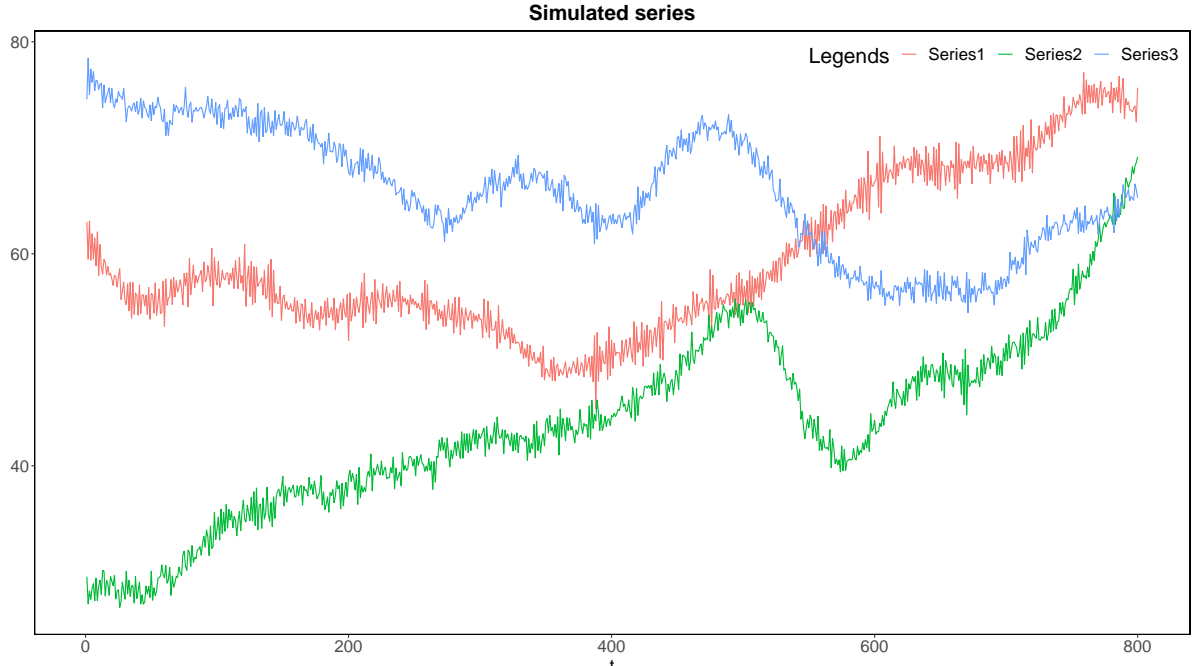

Figure 3. Simulated series from VAR(2) model with trend.

We used an LSTM network with one hidden layer of 20 units. The input at time t was $\mathbf{x}_t = \left(t, t^2, t^3, 1/t, 1/t^2, 1/t^3\right)'$. The learning rates were $\eta_1 = 0.001$ and $\eta_2 = 0.01$, with $K = 6,000$ iterations and precision $prec = 10^{-5}$.

4.1. Simulation results

Table 1 reports summary statistics of 100 estimates of each parameter, where $a_{j,k}^{(i)}$ refers to the (j,k)-th entry of the coefficient matrix A_i and $\sigma_{j,k}$ is the (j,k)-th entry of the variance-covariance

matrix Σ . We can observe that the estimated values are close to the true ones with small standard deviations (SDs) and biases.

Table 1. Estimation results: true value above sample mean, standard deviation of 100 estimates of each parameter and sample bias.

	$a_{1,1}^{(1)}$	$a_{1,2}^{(1)}$	$a_{1,3}^{(1)}$	$a_{2,1}^{(1)}$	$a_{2,2}^{(1)}$	$a_{2,3}^{(1)}$	$a_{3,1}^{(1)}$	$a_{3,2}^{(1)}$	$a_{3,3}^{(1)}$
True value	-1.0842	-0.1245	0.3137	-0.7008	-0.3754	-0.2064	0.3166	0.3251	0.2135
Mean	-1.0769	-0.1220	0.2768	-0.7057	-0.3754	-0.1941	0.2921	0.3471	0.2780
SD	0.0626	0.0765	0.1155	0.0693	0.1027	0.0795	0.0781	0.1005	0.1551
Bias	0.0073	0.0025	-0.0369	-0.0049	0.0000	0.0123	-0.0245	0.0220	0.0645
	$a_{1,1}^{(2)}$	$a_{1,2}^{(2)}$	$a_{1,3}^{(2)}$	$a_{2,1}^{(2)}$	$a_{2,2}^{(2)}$	$a_{2,3}^{(2)}$	$a_{3,1}^{(2)}$	$a_{3,2}^{(2)}$	$a_{3,3}^{(2)}$
True value	-0.5449	-0.3052	-0.1952	-0.4057	0.5129	0.3655	0.0054	-0.2911	0.2066
Mean	-0.5357	-0.2924	-0.2060	-0.4093	0.5123	0.3797	-0.0243	-0.2635	0.2642
SD	0.0947	0.1078	0.0575	0.1166	0.1310	0.0656	0.1081	0.0929	0.1147
Bias	0.0092	0.0128	-0.0108	-0.0036	-0.0006	0.0142	-0.0297	0.0276	0.0576
	$\sigma_{1,1}$	$\sigma_{2,1}$	$\sigma_{2,2}$	$\sigma_{3,1}$	$\sigma_{3,2}$	$\sigma_{3,3}$			
True value	0.4834	-0.2707	0.4079	0.1368	-0.0221	0.4103			
Mean	0.4871	-0.2723	0.4246	0.1251	-0.0380	0.4420			
SD	0.0502	0.0477	0.1829	0.0370	0.0755	0.0890			
Bias	0.0037	-0.0016	0.0167	-0.0117	-0.0159	0.0317			

Following (Fan et al., 2003), we use the mean absolute deviation

$$\mathrm{MAD} = \frac{1}{3 \times 800} \sum_{k=1}^{3} \sum_{t=1}^{800} \left| \hat{\mu}_{k,t}^{(i)} - \mu_{k,t} \right|$$

to evaluate the accuracy of trend estimation in the ith simulation run, $i = 1,...,100$. Figure 4 presents three sets of estimated trends with MAD at the first quartile (short dashed, black), the median (dotted, red), and the third quartile (long dashed, black) respectively among the 100 simulation runs, which follow the true trend very closely, closely and reasonably well.

Series1

Series2

Series3

Legends —— True value ······ Median --- First quartile --· Third quartile
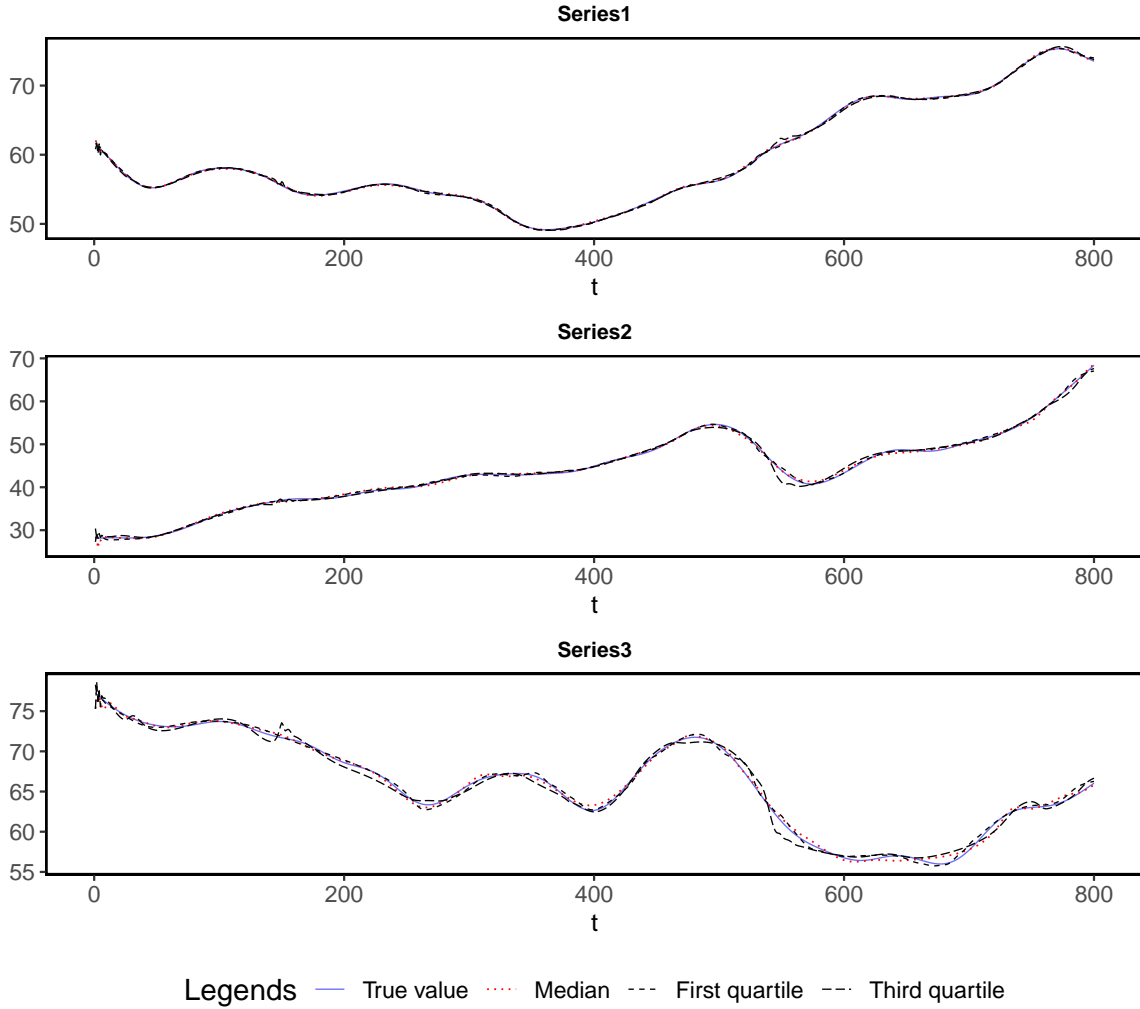
Figure 4. True (solid, blue) and estimated trends with MAD at first quartile (short dashed, black), third quartile (long dashed, black), and median (dotted, red).

5. Real data application

For the US macroeconomic series (Figure 1), we fit a model and make forecasts 20 times, each time using a training sample of size $T = 166$. The training samples are $y_{1:T}^{(i)} = \{\mathbf{y}_i, \mathbf{y}_{i+1}, ..., \mathbf{y}_{i+T-1}\}$, $i = 1, ..., 20$, and we forecast $h = 1, 2, ..., 8$ quarters ahead.

The first model fitted is DeepVARwT ($p = 2$). An LSTM network with a hidden layer of 20 units was used. The learning rates were $\eta_1 = 0.001$ and $\eta_2 = 0.01$, with $K = 9,800$ iterations and precision $prec = 2 \times 10^{-6}$. The estimated trends (red) are shown in Figure 5 for the first training sample ($i = 1$), which can be seen to follow the observations (blue) smoothly. Results for other training samples are available at `https://github.com/lixixibj/DeepVARwT-data-code/tree/main/real-data-forecast-res`.
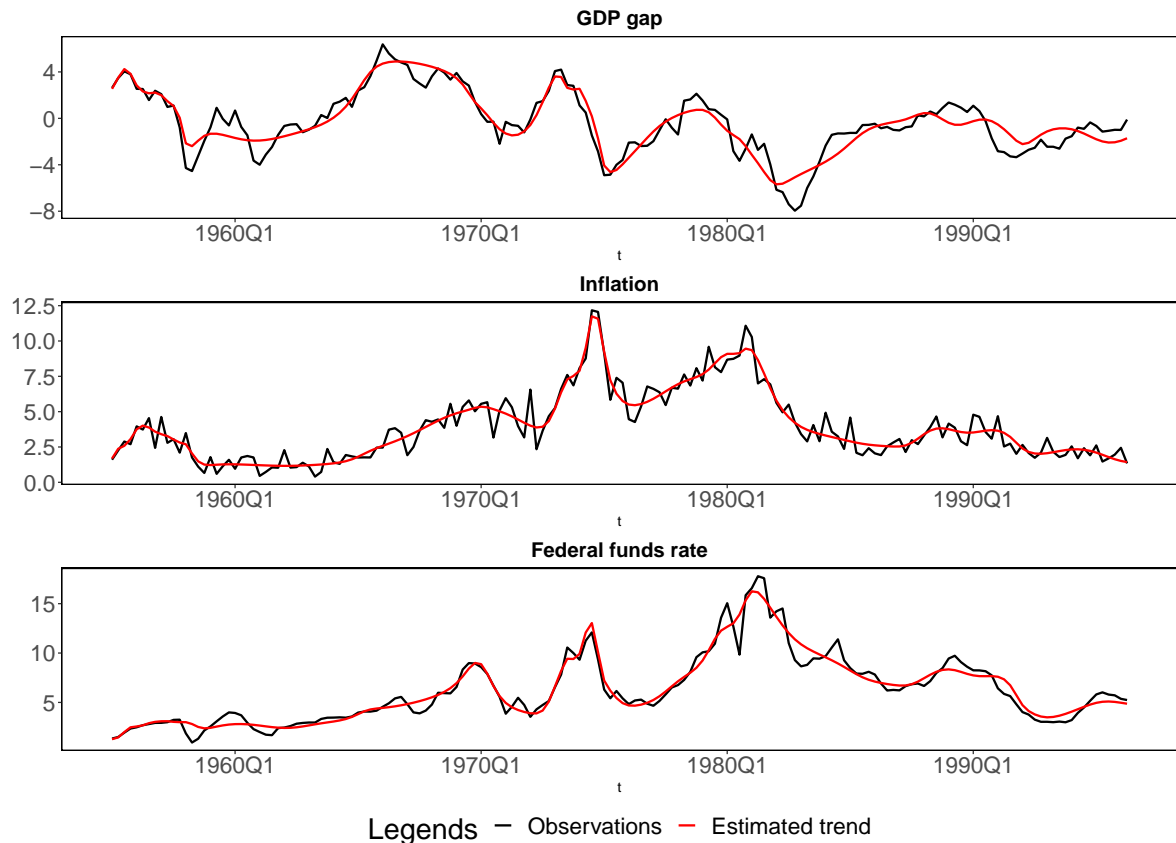
Figure 5. The first training sample (black lines) from 1955Q1 to 1996Q2 and the corresponding estimated trends (red lines).

The sample autocorrelations of residuals are shown in Figure 6. The results are very good for the GDP gap series, reasonable for the Inflation series, and a little concerning for the Federal funds rate series in terms of the number of values outside the boundaries.
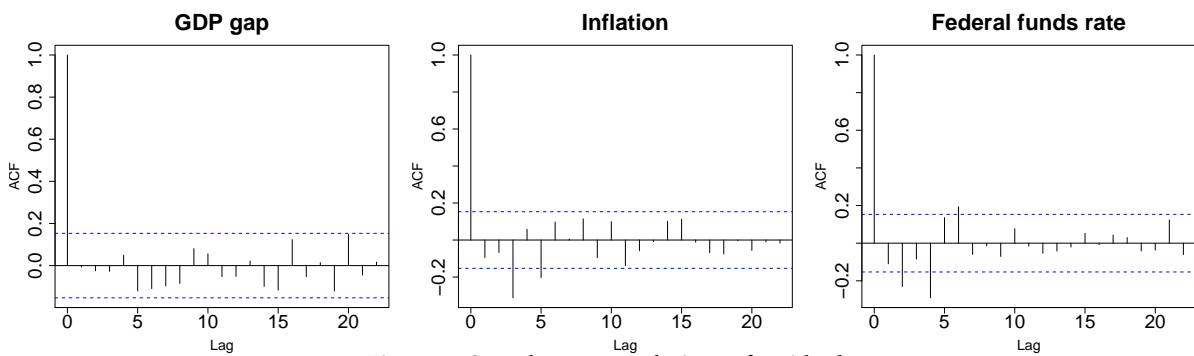

Figure 6. Sample autocorrelations of residuals.

Figure 7 contains normal QQ plots of the residuals. There is slight deviation from normality for the Federal funds rate series at both ends.
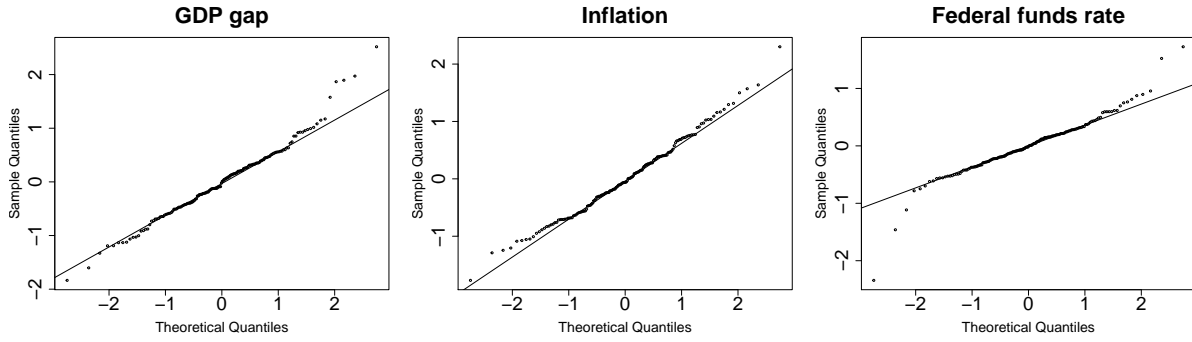
13

Figure 7. Normal QQ plots of residuals.

For comparison, we also fitted a VAR(2) model with cubic trend using exogenous variables $\mathbf{x}_t = (t, t^2, t^3)'$, a DeepAR and a DeepState model using default hyperparameter values. Table 2 gives a summary of these models and the software packages used.

Table 2. Models used for comparison.

Model	Description	Available software
VAR with trend	Vector autoregressive model with trend (Pfaff & Stigler, 2018)	`vars::VAR(exogen=x)`
DeepAR	Deep learning based autoregressive model (Salinas et al., 2020)	`gluonts.DeepAREstimator()`
DeepState	Deep learning based state space model (Rangapuram et al., 2018)	`gluonts.DeepStateEstimator()`

To evaluate the accuracy of point forecasts, we computed the h-step-ahead Absolute Percentage Error averaged over 20 forecasts

$$\text{APE}(h) = \frac{1}{20} \sum_{i=1}^{20} \left| \frac{y_{T+h}^{(i)} - \hat{y}_{T+h}^{(i)}}{y_{T+h}^{(i)}} \right| \times 100$$

for each component series $\{y_t^{(i)}\}$ in the ith training sample, where $\hat{y}_{T+h}^{(i)}$ is the h-step-ahead forecast of $y_{T+h}^{(i)}$. The Scaled Interval Score (Gneiting & Raftery, 2007) is averaged as follows:

$$\text{SIS}(h) = \frac{1}{20} \sum_{i=1}^{20} \frac{\left(u_{T+h}^{(i)} - l_{T+h}^{(i)}\right) + \frac{2}{\alpha}\left(l_{T+h}^{(i)} - y_{T+h}^{(i)}\right) \mathbb{1}_{\left\{y_{T+h}^{(i)} < l_{T+h}^{(i)}\right\}} + \frac{2}{\alpha}\left(y_{T+h}^{(i)} - u_{T+h}^{(i)}\right) \mathbb{1}_{\left\{y_{T+h}^{(i)} > u_{T+h}^{(i)}\right\}}}{\frac{1}{T-s} \sum_{t=s+1}^{T} \left|y_t^{(i)} - y_{t-s}^{(i)}\right|},$$

to measure the overall accuracy of the $(1 - \alpha) \times 100\%$ prediction intervals $(l_{T+h}^{(i)}, u_{T+h}^{(i)})$ for the ith training sample, $i = 1, ..., 20$, where $\mathbb{1}_A$ is the indicator function for the condition A, s is the seasonality of the time series ($s = 4$ for quarterly data).

Table 3 shows the forecasting performances of different models over several horizons $h = 1, 2, 4, 8$ and averages over $h = 1, ..., 4$ and $h = 1, ..., 8$. When a model performs best, the corresponding number in the table will be bold.

- **DeepVARwT vs VARwT.** Compared with VARwT, DeepVARwT produced better point forecasts at almost all forecasting horizons for the GDP gap (except for $h = 8$) and the Federal funds rate (except for $h = 1$) series. For the Inflation series, DeepVARwT performed better in long-term point forecasting ($h = 4$ and 8). DeepVARwT resulted in better prediction intervals at horizons up to $h = 4$ quarters for the GDP gap and the Federal funds rate series.

- **DeepVARwT vs other deep learning based models.** Compared with DeepAR and DeepState, our model gave more accurate point forecasts and prediction intervals at almost all the forecasting horizons for the GDP gap (except for $h = 8$) and the Federal funds rate series.

- **Summary.** Overall, the DeepVARwT model often produced better forecasts and prediction intervals than other models for the GDP gap and the Federal funds rate series. Its performance on the Inflation series is best for point forecasts over $h = 1 : 8$, but falls a little behind VARwT and DeepAR for prediction intervals over $h = 1 : 4$ and $h = 1 : 8$. The python code and data to reproduce forecasting results is available at `https://github.com/lixixibj/DeepVARwT-data-code`.

Table 3. Performance of DeepVARwT against other models according to APE and SIS.

	GDP gap											
	Absolute Percentage Error						Scaled Interval Score					
	h=1	h=2	h=4	h=8	h=1:4	h=1:8	h=1	h=2	h=4	h=8	h=1:4	h=1:8
VARwT	821.280	1174.785	171.781	223.257	582.674	394.507	1.710	2.444	4.009	**6.114**	2.820	**4.130**
DeepAR	275.052	404.041	153.950	**193.069**	240.014	215.141	9.518	14.813	25.361	50.041	17.893	30.634
DeepState	989.870	1065.755	181.548	207.751	604.521	403.498	8.496	16.941	23.140	35.239	17.193	25.595
DeepVARwT	**127.038**	**377.765**	**126.372**	211.567	**183.279**	**183.724**	**1.438**	**2.291**	**3.746**	14.867	**2.486**	6.285
	Inflation											
	Absolute Percentage Error						Scaled Interval Score					
	h=1	h=2	h=4	h=8	h=1:4	h=1:8	h=1	h=2	h=4	h=8	h=1:4	h=1:8
VARwT	**28.056**	35.047	50.148	105.575	38.183	61.539	**3.579**	**4.177**	4.963	6.259	**4.340**	**5.084**
DeepAR	29.578	**29.272**	51.156	49.654	**36.729**	45.070	8.493	8.938	12.126	13.863	9.709	11.259
DeepState	79.052	71.800	69.424	58.932	74.591	70.214	6.071	5.492	**3.842**	**5.363**	5.188	6.301
DeepVARwT	36.557	48.426	**50.027**	**36.364**	46.045	**43.356**	8.916	10.617	6.221	7.091	8.548	8.444
	Federal funds rate											
	Absolute Percentage Error						Scaled Interval Score					
	h=1	h=2	h=4	h=8	h=1:4	h=1:8	h=1	h=2	h=4	h=8	h=1:4	h=1:8
VARwT	**6.192**	13.971	40.117	77.885	21.623	43.246	2.202	3.295	4.573	**5.361**	3.532	**4.351**
DeepAR	13.265	25.071	57.983	116.748	34.298	64.475	4.604	13.674	33.545	48.941	18.973	29.549
DeepState	34.429	39.658	57.774	133.894	45.139	72.657	6.825	9.882	12.488	24.072	10.151	15.405
DeepVARwT	8.530	**11.813**	**28.709**	**77.856**	**16.771**	**37.386**	**2.063**	**2.282**	**4.062**	8.301	**2.876**	4.964

6. Concluding remarks

In this work, we proposed a new approach to VAR modeling and forecasting by generating trends as well as model parameters using an LSTM network and the associated deep learning

methodology for exact maximum likelihood estimation. A simulation study and a real data example demonstrated the effectiveness of the proposed approach.

7. Acknowledgements

References

Ansley, C. F., & Kohn, R. (1986). A note on reparameterizing a vector autoregressive moving average model to enforce stationarity. *Journal of Statistical Computation and Simulation*, *24*, 99–106.

Barndorff-Nielsen, O., & Schou, G. (1973). On the parametrization of autoregressive models by partial autocorrelations. *Journal of Multivariate Analysis*, *3*, 408–419.

de Bézenac, E., Rangapuram, S. S., Benidis, K., Bohlke-Schneider, M., Kurle, R., Stella, L., Hasson, H., Gallinari, P., & Januschowski, T. (2020). Normalizing kalman filters for multivariate time series analysis. *Advances in Neural Information Processing Systems*, *33*.

Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, *12*.

Fan, J., Yao, Q., & Cai, Z. (2003). Adaptive varying-coefficient linear models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *65*, 57–80.

Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to forget: Continual prediction with LSTM. *Neural Computation*, *12*, 2451–2471.

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

Graves, A., Liwicki, M., Fernández, S., Bertolami, R., Bunke, H., & Schmidhuber, J. (2008). A novel connectionist system for unconstrained handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *31*, 855–868.

Hannan, E. J. (1970). *Multiple time series* volume 38. John Wiley & Sons.

Heaps, S. E. (2022). Enforcing stationarity through the prior in vector autoregressions. *Journal of Computational and Graphical Statistics*, (pp. 1–24).

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*, 1735–1780.

Jordà, Ò. (2005). Estimation and inference of impulse responses by local projections. *American Economic Review*, *95*, 161–182.

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

Morf, M., Vieira, A., Kailath, T. et al. (1978). Covariance characterization by partial autocorrelation matrices. *The Annals of Statistics*, *6*, 643–648.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., & Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 32* (pp. 8024–8035). Curran Associates, Inc. URL: `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

Pfaff, B., & Stigler, M. (2018). *vars: VAR Modelling*. URL: `https://cran.r-project.org/web/packages/vars/index.html` r package version 1.5-3.

Rangapuram, S. S., Seeger, M. W., Gasthaus, J., Stella, L., Wang, Y., & Januschowski, T. (2018). Deep state space models for time series forecasting. In *Advances in Neural Information Processing Systems* (pp. 7785–7794).

Rezende, D., & Mohamed, S. (2015). Variational inference with normalizing flows. In *International Conference on Machine Learning* (pp. 1530–1538). PMLR.

Roy, A., McElroy, T. S., & Linton, P. (2019). Constrained estimation of causal invertible VARMA. *Statistica Sinica*, *29*, 455–478.

Sak, H., Senior, A. W., & Beaufays, F. (2014). Long short-term memory recurrent neural network architectures for large scale acoustic modeling, .

Salinas, D., Bohlke-Schneider, M., Callot, L., Medico, R., & Gasthaus, J. (2019). High-dimensional multivariate forecasting with low-rank gaussian copula processes. In *Advances in Neural Information Processing Systems* (pp. 6827–6837).

Salinas, D., Flunkert, V., Gasthaus, J., & Januschowski, T. (2020). DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, *36*, 1181–1191.

Wand, M. P., & Jones, M. C. (1994). *Kernel smoothing*. CRC Press.

Wang, Y., Smola, A., Maddix, D., Gasthaus, J., Foster, D., & Januschowski, T. (2019). Deep factors for forecasting. In *International Conference on Machine Learning* (pp. 6607–6617). PMLR.

Appendix A. Prediction error variances and covariances

First consider the model for $\{\mathbf{y}_t\}$ to be VAR(1) with trend:

$$\mathbf{y}_t - \boldsymbol{\mu}_t = A(\mathbf{y}_{t-1} - \boldsymbol{\mu}_{t-1}) + \boldsymbol{\varepsilon}_t, \tag{A.1}$$

where $\{\boldsymbol{\varepsilon}_t\}$ is a white noise, $\boldsymbol{\varepsilon}_t \sim \mathcal{N}(\mathbf{0}, \Sigma)$ and $\boldsymbol{\varepsilon}_t$ is uncorrelated with $\mathbf{y}_{t-1}, \mathbf{y}_{t-2}, \dots$

Then, we can decompose $\mathbf{y}_{T+\ell}$ starting with $\mathbf{y}_T$ for $\ell = 1, \dots, h$:

$$\mathbf{y}_{T+1} - \boldsymbol{\mu}_{T+1} = A(\mathbf{y}_T - \boldsymbol{\mu}_T) + \boldsymbol{\varepsilon}_{T+1},$$

$$\mathbf{y}_{T+2} - \boldsymbol{\mu}_{T+2} = A(\mathbf{y}_{T+1} - \boldsymbol{\mu}_{T+1}) + \boldsymbol{\varepsilon}_{T+2} = A^2(\mathbf{y}_T - \boldsymbol{\mu}_T) + A\boldsymbol{\varepsilon}_{T+1} + \boldsymbol{\varepsilon}_{T+2},$$

$$\mathbf{y}_{T+3} - \boldsymbol{\mu}_{T+3} = A^3(\mathbf{y}_T - \boldsymbol{\mu}_T) + A^2\boldsymbol{\varepsilon}_{T+1} + A\boldsymbol{\varepsilon}_{T+2} + \boldsymbol{\varepsilon}_{T+3}, \tag{A.2}$$

$$\vdots$$

$$\mathbf{y}_{T+h} - \boldsymbol{\mu}_{T+h} = A^h(\mathbf{y}_T - \boldsymbol{\mu}_T) + \sum_{i=0}^{h-1} A^i \boldsymbol{\varepsilon}_{T+h-i},$$

where A^i is understood to be the identity matrix when $i = 0$.

From (A.2), the best linear predictor for $\mathbf{y}_{T+\ell}$ given $\mathbf{y}_T, \mathbf{y}_{T-1}, \dots$ is

$$\hat{\mathbf{y}}_{T+\ell} = E\left[\mathbf{y}_{T+\ell} \mid \mathbf{y}_T, \mathbf{y}_{T-1}, \dots\right] = A^\ell(\mathbf{y}_T - \boldsymbol{\mu}_T) + \boldsymbol{\mu}_{T+\ell}, \tag{A.3}$$

and the associated prediction error variance-covariance matrix is

$$Var[\mathbf{y}_{T+\ell} - \hat{\mathbf{y}}_{T+\ell}] = \sum_{i=0}^{\ell-1} A^i \Sigma (A^i)'. \tag{A.4}$$

When $\{\mathbf{y}_t\}$ follows the VAR(p) model (3) with trend, we use its VAR(1) form

$$\mathbf{y}_t^* - \boldsymbol{\mu}_t^* = A^*(\mathbf{y}_{t-1}^* - \boldsymbol{\mu}_{t-1}^*) + \boldsymbol{\varepsilon}_t^*, \tag{A.5}$$

where $\mathbf{y}_t^* = (\mathbf{y}_t', \mathbf{y}_{t-1}', \dots, \mathbf{y}_{t-p+1}')'$, $\boldsymbol{\mu}_t^* = (\boldsymbol{\mu}_t', \boldsymbol{\mu}_{t-1}', \dots, \boldsymbol{\mu}_{t-p+1}')'$,

$$A^* = \begin{bmatrix} A_1 & A_2 & \cdots & \cdots & A_p \\ I & O & \cdots & \cdots & O \\ O & I & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ O & \cdots & O & I & O \end{bmatrix}, \tag{A.6}$$

and

$$\mathbf{y}_t = [I, \mathbf{O}, \cdots, \mathbf{O}]\mathbf{y}_t^*. \tag{A.7}$$

The variance-covariance matrix of $\varepsilon_t^* = (\varepsilon_t', \mathbf{0}', ..., \mathbf{0}')'$ is

$$\Sigma^* = \begin{bmatrix} \Sigma & \mathbf{O} \\ \mathbf{O} & \mathbf{O} \end{bmatrix}. \tag{A.8}$$

Using (A.3), the best linear predictor for $\mathbf{y}_{t+h}^*$ given $\mathbf{y}_t^*, \mathbf{y}_{t-1}^*, ...$ is

$$\hat{\mathbf{y}}_{t+h}^* = (\mathbf{A}^*)^h(\mathbf{y}_t^* - \boldsymbol{\mu}_t^*) + \boldsymbol{\mu}_{t+h}^*. \tag{A.9}$$

Using (A.4), the variance-covariance matrix of the prediction error for $\hat{\mathbf{y}}_{t+h}^*$ is

$$\sum_{i=0}^{h-1} (\mathbf{A}^*)^i \Sigma^* ((\mathbf{A}^*)^i)'. \tag{A.10}$$

The prediction for $\mathbf{y}_{t+h}$ can be extracted from that for $\mathbf{y}_{t+h}^*$. The prediction error variance-covariance matrix for $\hat{\mathbf{y}}_{t+h}$ is in the top-left corner of the above.