Deep Vector Autoregression for Macroeconomic Data

Marc Agustí (marc.agusti@barcelonagse.eu)
Patrick Altmeyer (patrick.altmeyer@barcelonagse.eu)
Ignacio Vidal-Quadras Costa (ignacio.vidalquadrascosta@barcelonagse.eu)

junio, 2021

Abstract

Vector autoregression models have been the traditional technique in the last decades when it comes to the forecasting of time series data. In particular, the VAR framework has demonstrated its outperformance in precision and accuracy of predicting the next lags of time series, hence becoming a useful tool for policy makers, whcih rely on this methodology to construct forecasts or even for the analysis of impulse response functions. However, with the recent advancement in computational power of computers, and more importantly, the development of more advanced machine learning algorithms and approaches such as deep learning, new algorithms are developed to analyze and forecast time series data. This paper aims to contribute to the time series literature by using the Long Short-Term Memory (LSTM) to see whether this new advanced methodology is superior to the usual VAR framework. By fitting each regression of the VAR with a neural network instead of with a simple OLS regression, we are able to outperform both in-sample and out-of-sample the predictions of the usual VAR for variables such as GDP, FED funds rate and inflation, ...

Keywords— Vector Autoregression, Deep Learning, Neural Networks, Macroeconomic Timeseries

1 Introduction

As stated by the European Central Bank, the monetary transmission mechanism is the process through which monetary policy decisions affect the economy in general and the price level in particular. However, the uncertainty of this tramission is huge, given that this transmission is characterized by long, variable and uncertain time lags. Hence, it is difficult to predict how changes in monetary policy actions affect economic real variables. Therefore, it is of foremost importance for policy-makers to come up with tools that allow them to predict these effects.

With this in mind, a lot of research on the forecasting of time series has been developed to asses the effect of current policy decisions on future economic variables. Thanks to this, over the last decades policy makers have had more information when taking decisions. This information can be of the form of point estimates and interval forecasts. To come up with these estimates, several methodologies have been largely applied in the time series forecasting literature.

Up until now, the most common methodology to come up with these estimates is the so-called Vector Autoregression (VAR). This framework, which belongs to the traditional class of econometric forecasting techniques, has proved to provide policy-makers with fairly good and consistent point and interval estimates. It has therefore been used in the monetary policy divisons of institutions such as the European Central Bank or the International Monetary Fund.

Simultaneously, with the recent advancement in computational power of computers, and more importantly, the development of more advanced machine learning algorithms and approaches such as deep learning, new algorithms are developed to analyze and forecast time series data. Whereas the good performance of techniques such as VAR is well-known, it is still uncertain whether deep learning algorithms can actually improve these forecasts.

To this end, this paper aims to provide with a new and ground-breaking methodology that combines the VAR equation-by-equation structure with the deep neural network architecture to capture the potential non-linear relationships in the data generating process. Thereupon, we construct what we called a Deep-VAR framework to generate point and interval forecasts. The ultimate objective of this empirical exercise is to provdie the existing literture with a method that outperforms the most conventional methods, yet keeping interpretability in the underlying mechanism.

To the best of our knowledge, this is the first paper to fit a deep neural network for each equation of the VAR structure. Although a lot of authors have used deep learning to forecast time series, all of them fit the whole system in one big neural network. We believe that by doing it equation-by-equation not only do we mantain interpretabiliy of the model but also we get better forecasting results.

We find that the Deep-VAR methodology outperforms the traditional VAR framework both in-sample and out-pof-sample. When it comes to forecasts, we also obtain a lower RMSE.

The rest of the paper is structured as follows: in section 2 we present a literature review of prior research on the methodologies used to provide forecasts and on the monetary trasmission mechanism. Section 3 is a detailed description of the data we use in our empirical exercise. In section 4 we present the traditional VAR methodology and we also present our Deep-VAR model following by our empirical findings in section 5. Finally, in section 6 we finish with concluding remarks.

2 Literature review

There is a large agreement among economists on the fact that monetary policy has a short-term influence on the economic activity. Friedman and Schwartz (2008) found that monetary policy actions are followed by movements in real output that may last for two years or more (Romer and Romer (1989); Bernanke (1990)). However, what are the forces that trigger this effect is of interest for most economists, in particular, economists aim to understand the monetary transmission mechanism. If monetary policy affects the real economy, what is the transmission mechanism by which these effects occur? This is one of the questions which is among the most important and controversial in macroeconomics.

In the aftermath of the oil price shock in the 1970's, interest was raised in understanding business cycles. To do that, most economists made use of large-scale macroeconomic models, which was criticized by Lucas Jr (1976), stating that the assumption of invariant behavioral equations was inconsistent with the dynamic maximizing behavior. Hence, New Classical economists started making use of the so-called market clearing models of economic fluctuations. With the goal of really taking into account productivity shocks, Real Business Cycle models were developed (Kydland and Prescott (1982)).

After the failure of the large-scale macroeconomic models when trying to predict business cycles, the economic profession tried to solve this by means of the use of structural vector autoregression (VAR) models to analyze business cycles, which were useful to capture the impact of policy-actions. Sims and others (1986) suggested that VARs were useful to evaluate macroeconomic models. One of the advantage of VARs is that they are not a large and complicated structure, and hence are easily interpretable (do not suffer from the "black box" problem).

In the last decades the use of VAR's in order to do time series forecasting has been quite extensive. Actually, a lot of different models have been proposed with the intention to model and predict time series data. When it comes to the VAR framework, the different factors in the projected VAR models are difficult to understand, and that is why researchers rely heavily on impulse response functions (IRF) (Enders (2008)).

As for now, the models we have seen are more classical econometric based models that are not able to capture nonlinear relationships in the data, which might be sometimes a limitation. In the case of economic time series, specifically gdp, inflation and so on, nonlinear trends are likely to appear and be present in the essence of the data generating process as shown by Brock et al. (1991).

In the past years, authors have therefore started using nonlinear techniques for forecasting. Machine Learning has contributed a lot to this field. The most popular machine learning techniques which do not assume a linear relationship between inputs and outputs are K-Nearest Neighbors (first introduced by Fix and Hodges (1951)), Support Vector Machines (mostly developed by Cortes and Vapnik (1995)), Random Forests (first introduced in 1995 by Ho (1995)) and Neural Networks (NN) (first proposed in 1943 by McCulloch and Pitts (1990)).

The most recent and complex algorithm of the ones mentioned above is Neural Networks (NN). NN are a relatively new nonlinear technique to which a lot of authors have made contributions. One of the main advantages of NN compared to the linear models is that they can approximate any nonlinear functions without any apriori information about the properties of the data series. In our case we are interested in the application of NN to time series.

The main idea of NN is to reproduce the inner working of the human mind. NN consist of thousands or even millions of simple processing neurons that are densely interconnected. These neurons are

organized in layers. A NN contains an input layer, one or more hidden layers and an output layer and information flows from one layer to another using weight. One neuron receives information form other neurons in the previous layer and passes the processed information into the next layer. This information can flow only in one direction (from the previous layer to the next layer), which means that the NN is "feed-forward" or can retrieve information. Each connection between neurons is weighted. When the network is active, the neuron receives a different data (information) from each of its connections and multiplies it by the associated weight. It then adds the resulting products together and passes this output through an activation function that maps the output, yielding the final output. Finally, this output is then passed onto the next neurons. This processed is repeated until we reach the output layer.

With this new algorithm coming into play, Zhang, Patuwo, and Hu (1998) used NN for forecasting. Recently, artificial neural networks (ANN) have played attention enhancing devotions in the field of time series predicting (Hamzaçebi (2008), Zhang (2003), Kihoro, Otieno, and Wafula (2004)). In particular, ANNs have the advantage of not assuming the statistical distribution followed by the values, being able of proficiently capturing non-linearities. That is, they are self-adaptive (Zhang, Patuwo, and Hu (1998), Zhang (2003)).

A class of ANN is the called recurrent neural network (RNN). RNN allows to use previous outputs as inputs, this allows the model to retain information about the past, making it very efficient for time series. This has been shown by Dorffner (1996). In this article, the author highlights the power of RNN for forecasting compared with standard linear models.

For this reason, a lot of authors interested in forecasting economic series have compared linear models with nonlinear models. In economic time series, in the short run, the series is expected to behave more or less the same way it has been behaving up to this point, but on the other hand, if we are interested in forecasting at a big window, then then is when chaos and instability appear, meaning that nonlinear relations may arise, making it more appealing to use ANN as they are capable of identifying these turning points as they do not assume a linear relationship of inputs and outputs.

This was shown by a recent paper of the Bank of England, Joseph et al. (2021). In this paper they run a horse race for forecasting inflation among different horizons comparing the performance of linear and nonlinear algorithms. The results support out hypothesis that NN and other nonlinear Machine Learning algorithms are useful for forecasting at a longer horizon given that, the longer the horizon, the more likely it is to find the chaos and instability Brock et al. (1991) talked about. And, as previously exposed, these turning points are hard to spot with linear relationships of inputs and outputs, while they might be easier to spot with nonlinear models, like SVM or NN.

One of the problems of RNN is the long-term dependency. To illustrate it with an example, some decisions are made taking into account information that happened way back in the past. RNN struggle to keep this very old information threfore limiting its forecasting power. In order to solve this problem, Hochreiter and Schmidhuber (1997) introduced the LSTM in the paper Long Short-Term Memory. This is the reason why a lot of authors use this type of RNN when forecasting any kind of time series.

Yet, when we are interested in the monetary transition mechanism, we are not just interested in the forecasting accuracy of the model we are use but we are also interested in the inference. Central banks need to know if rates granger cause one variable or not or they also need to know the IRF in order to implement the correct monetary policy.

The linear additive relationship of linear models allows the model to observe which variable granger

cause another and what are the IRF of each variable with respect to another. Unfortunately, in the case of nonlinear models, its nature makes it impossible to recover these insights because the outputs and the inputs do not have a linear additive structure.

Therefore, on the one hand, the nonlinear structure of NN helps us for forecasting in the case that there are some nonlinear relationships in the series, but on the other hand we lose the interpretability of the model, making it impossible to recover IRF or to even to assess if one variable granger causes another. This problem is also known as the black box problem. This is because the data fed into the input layer passes through the succeeding layers, getting multiplied, added together and transformed in complex and different ways, until it finally arrives, radically transformed, at the output layer. Therefore, it is impossible to assess what happened with one input and how it affected the output.

Whether neural networks can do the job that so far has been done by VARs (and its extensions) is not so clear. In the past years some part of the research in this field has been put towards the study of neural networks for time series modelling. The research question investigated in this article is that whether and how the newly developed deep learningbased algorithms for forecasting time series data, such as LSTM are superior to the traditional algorithms such as VAR. Therefore, assessing the accuracy of forecasts is necessary when employing various forms of forecasting methods, and more specifically forecasting using regression analysis as they have several limitations in applications

3 Data

To study the monetary transmission mechanism we used balncaded panel of monthly US data which spans the period of January 1959 through March 2021. In order to explain the monetary transmission mechanism, the VAR literature relies on different indicators such us output and income, industrial new orders and turnover, retail sales and turnover, building permits, employment, consumption, price indices, exchange rates, short- and long-term interest rates, stock price indices, money and credit quantity aggregates, balance of payments and external trade, confidence indicators, and some foreign variables such as output, prices, interest rates, and stock markets from other countries used as proxies for external real, nominal and monetary influences.

On this paper we relied on four of the macroeconomic indicators mentioned above; output, price indices, employment and rates. For each indicator we chose a time series as its proxy. We can see the proxies used for each variable below.

- *output*: Industrial Production (IP)
- *price indices*: Consumer Price Index (CPI)
- *unemployment*: Unemployment rate (UR)
- *rates*: US Federal Fund Rates (FFR)

Note that we use IP rather than GDP as a proxy for output because for the case of the GDP only quartery data was available.

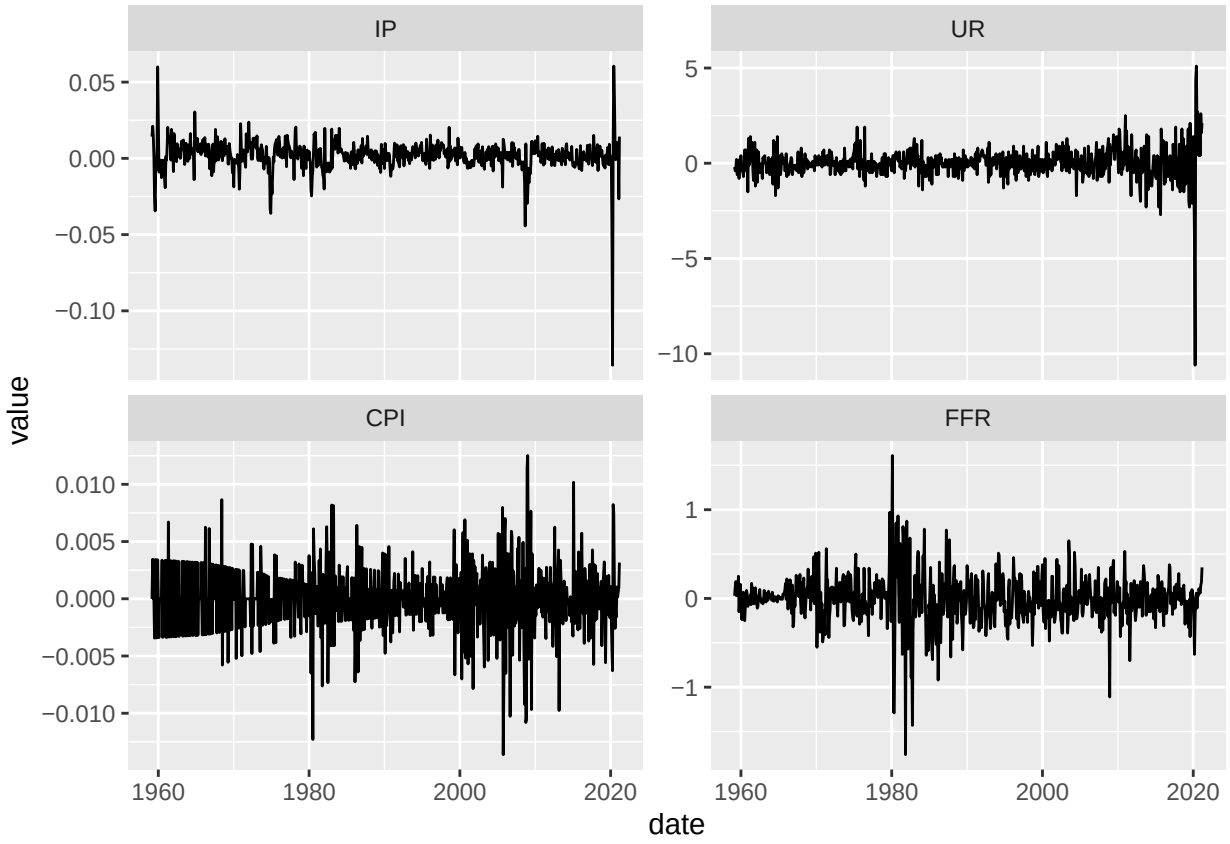
As consequence of working with monthly data, we worked 745 observations, each with the four variables previously mentioned.

The data is taken from the Economic Research department of the Federal Reserve Bank of Saint Louis (FRED). In a recent paper, they have created a monthly database for economic research

(FRED-MD), a macroeconomics data base which consists of 134 monthly US indicators. The data is automatically updated each month and can be downloaded freely, facilitating the replication of results and comparing results.

Another strength of using the FRED-MD is the fact that the data is already preprocessed. To see how it has been preprocessed see the Methodology section.

In the graph below we can find the time series already preprocessed by the FRED-MD.



	IP	UR	CPI	FFR
min	-0.14	-10.60	-0.01	-1.76
max	0.06	5.10	0.01	1.61
mean	0.00	0.02	0.00	-0.00
sd	0.01	0.81	0.00	0.27
var	0.00	0.66	0.00	0.08
median	0.00	0.00	-0.00	0.00
IQR	0.01	0.70	0.00	0.28

Table 1: Summary statistics

4 Pre Covid vs Post Covid Summary Stats

	IP	UR	CPI	FFR
min	-0.04	-2.70	-0.01	-1.76
max	0.06	2.50	0.01	1.61
mean	0.00	0.01	0.00	-0.00
sd	0.01	0.63	0.00	0.27
var	0.00	0.40	0.00	0.08
median	0.00	0.00	-0.00	0.00
IQR	0.01	0.63	0.00	0.28

Table 2: Summary statistics pre Covid

	IP	UR	CPI	FFR
min	-0.14	-10.60	-0.01	-0.63
max	0.06	5.10	0.01	0.35
mean	-0.00	0.68	0.00	0.01
sd	0.05	4.04	0.00	0.23
var	0.00	16.31	0.00	0.05
median	0.01	1.60	0.00	0.06
IQR	0.01	2.20	0.00	0.10

Table 3: Summary statistics post Covid

5 Methodology

In conventional Vector Autoregression (VAR) dependencies of any system variable on past realizations of itself and its covariates are modelled through linear equations. This corresponds to a particular case of the broader class of Deep Vector Autoregressions investigated here and will serve as the baseline for our analysis.

5.1 Vector Autoregression

Let $\mathbf{y}_t$ denote the $(K \times 1)$ vector of variables at time t. Then the VAR(p) with p lags and a constant deterministic term is simply a linear system of stochastic equations of the following form:

$$\mathbf{y}_t = \mathbf{c} + \mathbf{A}_1\mathbf{y}_{t-1} + \mathbf{A}_2\mathbf{y}_{t-2} + ... + \mathbf{A}_p\mathbf{y}_{t-p} + \mathbf{u}_t \tag{1}$$

The matrices $\mathbf{A}_m$, $m \in \{1, ..., p\}$ contain the reduced form coefficients and $\mathbf{u}_t$ is a vector of errors for which $\mathbb{E}\mathbf{u}_t$, $\mathbb{E}\mathbf{u}_t\mathbf{u}_t^T = \Sigma$ and $\mathbb{E}\mathbf{u}_t\mathbf{u}_s^T = \mathbf{0}$ for all $t \neq s$. We refer to (1) as the **reduced form** representation of the VAR(p) because all right-hand side variables are predetermined (Kilian and Lütkepohl 2017).

We can restate (1) more compactly as

$$\mathbf{y}_t = \mathbf{A}\mathbf{Z}_{t-1} + \mathbf{u}_t \tag{2}$$

where $\mathbf{A} = (\mathbf{c}, \mathbf{A}_1, \mathbf{A}_2, ..., \mathbf{A}_p)$ is of dimension $(K \times (Kp + 1))$ and $\mathbf{Z}_{t-1} = (1, \mathbf{y}_{t-1}^T, ..., \mathbf{y}_{t-p}^T)^T$ is of dimension $((Kp + 1) \times 1)$. The expression in (2) demonstrates that the VAR(p) can be considered as a **seemingly unrelated regression** (SUR) model composed of individual regressions with common regressors (Greene 2012). In fact, it is useful to note for our purposes that the VAR(p) can be estimated efficiently through equation-by-equation OLS regression. In particular, it follows from (2) that

$$y_{it} = c_i + \sum_{m=1}^{p} \sum_{j=1}^{K} a_{jm} y_{jt-m} + u_{it} \quad , \quad \forall i = 1, ..., K \tag{3}$$

which corresponds to the key modelling assumption that at any point in time t any time series $i \in 1, ..., K$ is just a weighted sum of past realizations of itself and all other variables in the system. This assumption makes the estimation VAR(p) processes remarkably simple. Perhaps more importantly, the assumption of linearity also greatly facilitates inference about VARs.

For implementation purposes it is generally more useful to estimate the VAR(p) through one single OLS regression. To this end let $\tilde{\mathbf{A}} = \mathbf{A}^{-1}$ and note that (2) can be restated even more compactly as

$$\mathbf{y} = \mathbf{Z}\tilde{\mathbf{A}} + \mathbf{u}_t \tag{4}$$

with $\mathbf{y} = (\mathbf{y}_1, ..., \mathbf{y}_T)^T$ $(T \times K)$ and $\mathbf{Z}$ $(T \times (Kp + 1))$. Then the closed form solution for OLS is simply $\tilde{\mathbf{A}} = (\mathbf{Z}^T\mathbf{Z})^{-1}\mathbf{Z}^T\mathbf{y}$ and hence

$$\mathbf{A} = \mathbf{y}^T\mathbf{Z}(\mathbf{Z}\mathbf{Z}^T)^{-1} \tag{5}$$

5.2 Deep Vector Autoregression

We propose the term Deep Vector Autoregression to refer to the broad class of Vector Autoregressive models that use deep learning to model the dependences between system variables through time. In particular, as before we let $\mathbf{y}_t$ denote the $(K \times 1)$ vector that describes the state of system at time t. Consistent with the conventional VAR we assume that each individual time series y_{it} can be modelled as a function of lagged realizations of all variables y_{jt-p}, $j = 1, ..., K$, $m = 1, ..., p$. More specifically we have

$$y_{it} = f_i\left(\mathbf{y}_{t-1:t-p}; \theta\right) + v_{it} \quad , \quad \forall i = 1, ..., K \tag{6}$$

where $\mathbf{y}_{t-1:t-p} = \{y_{jt-m}\}_{j=1,...,K}^{m=1,...,p}$ is the vector of lagged realizations, f_i is a variable specific mapping from past lags to the present and θ is a vector of parameters. While in the conventional VAR above we assumed that the multivariate process can be modelled as a system of linear stochastic equations, our proposed Deep VAR(p) can similarly be understood as a system of potentially highly non-linear equations. As we argued earlier, Deep Learning has been shown to be remarkably successful at learning mappings of arbitrary functional form (Goodfellow, Bengio, and Courville 2016).

Note that the input and output dimensions in (6) are exactly the same as in the conventional VAR(p) model (equation (3)): f_i maps from $\mathbf{y}_{t-1:t-p} \in \mathbb{R}^{Kp}$ to a scalar. Our proposed plain-vanilla

approach to Deep VARs diverges as little as possible from the conventional approach: it boils down to simply modelling each of the univariate outcomes in (6) as a deep neural network. We can restate this approach more compactly as

$$\mathbf{y}_t = \mathbf{f}(\mathbf{y}_{t-1:t-p}; \theta) + \mathbf{v}_t \tag{7}$$

where $\mathbf{f}(\cdot) = (f_1(\cdot), f_2(\cdot), ..., f_K(\cdot)^T$ is just the stacked vector of mappings to univariate outcomes described in (6).

The notation in (7) gives rise to a more unified and general approach to Deep VARs that would treat the whole process as one single dynamical system to be modelled through one deep neural network f:

$$\mathbf{y}_t = f(\mathbf{y}_{t-1:t-p}; \theta) + \mathbf{v}_t \tag{8}$$

This approach is in fact proposed and investigated by Verstyuk (2020) in his upcoming publication. We decided to go with the approach in (7) for two reasons: firstly, the link to conventional VARs is made abundantly clear through this implementation and, secondly, we found that the equation-by-equation approach produces good modelling outcomes and is relatively easy to implement using state-of-the art software.

Finally, note that if f_i in (3) is assumed to be linear and additive for all $i = 1, ..., K$ then we are back to the conventional VAR(p). This illustrates the point we made earlier that the linear VAR(p) is just a particular case of a Deep VAR(p). Since the model described in equations (6) and (7) is less restrictive but otherwise consistent with the conventional VAR framework, we expect that it outperforms the traditional approach towards modelling multivariate time series processes.

5.3 Deep Neural Networks - a whistle-stop tour

So far we have been speaking about deep learning in rather general terms. For example, above we have referred to our model of choice for learning the mapping $f_i : \mathbf{y}_{t-1:t-p} \mapsto y_{it}$ as a **deep neural network**. The class of deep neural networks can further be roughly divided into **feedforward neural networks** and **recurrent neural networks**. As the term suggests, the latter is generally used for sequential data and therefore our preferred model of choice. Nonetheless, below we will begin by briefly exploring feedforward neural networks first. This should serve as a good introduction to neural networks more generally and (even though we have not tested this empirically) there is good reason to believe that even Deep VARs using feedforward neural networks perform well.

5.3.1 Deep Feedforward Neural Networks

The term **deep feedforward neural network** or **multilayer perceptron** (MLP) is used to describe a broad class of models that are composed of possibly many functions that together make up the directed acyclical graph. The functions $f_i(\cdot)$ - sometimes referred as layers $\mathbf{h_i}$ - are chained together hierarchically with the first layer feeding forward its outputs to the second layer and so on (Goodfellow, Bengio, and Courville 2016). Applied to our case, an MLP with H hidden layers can be loosely defined as follows:

$$f_i(\mathbf{y}_{t-1:t-p};\theta) = f_i^{(H)}\left(f_i^{(H-1)}\left(...f_i^{(1)}\left(\mathbf{y}_{t-1:t-p}\right)\right)\right) \tag{9}$$

The depth of the MLP is defined by the number of hidden layers H where generally speaking deeper networks are more complex.

Need a reference here.

The desired outputs of any $f_i^{(h)}$ that will serve as inputs for $f_i^{(h+1)}$ cannot be inferred from the training data $\mathbf{y}_{t-1:t-p}$ ex-ante, which is where the term **hidden** layer stems from. Each $f_i^{(h)}$ is typically valued on a vector of hidden units, each of them receiving a vector of inputs from $f_i^{(h-1)}$ and returning a scalar that is referred to as activation value. This approach is inspired by neuroscience, hence the term **neural** network (Goodfellow, Bengio, and Courville 2016).

5.3.2 Deep Recurrent Neural Networks

Recurrent neural networks (RNN) are based on the idea of persistent learning: a continuous process that evolves gradually and at each step uses information about its prior states instead of continuously reinventing itself and starting from scratch. To this end, RNNs develop the basic concepts underlying feedforward neural networks by incorporating feedback loops. Formally the loop is typically made explicit as follows

$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t; \theta) \tag{10}$$

where $\mathbf{h}_t$ corresponds to the hidden state of the dynamical system at time t that the RNN learns (Goodfellow, Bengio, and Courville 2016). In the given context we have that $\mathbf{x}_t = \mathbf{y}_{t-1:t-p}$ as specified in (7). Given some random initial hidden state vector $\mathbf{h}_0$ the RNN updates parameters sequentially at each time step t as follows

$$\begin{aligned} \mathbf{a}_t &= \mathbf{b} + \mathbf{W}\mathbf{h_{t-1}} + \mathbf{U}\mathbf{h_{-1}} \\ \mathbf{h_t} &= \tanh(\mathbf{a}_t) \\ \mathbf{\hat{y}_t} &= \mathbf{c} + \mathbf{V}\mathbf{h_t} \end{aligned} \tag{11}$$

where $\mathbf{b}$ and $\mathbf{c}$ are vectors of constants (biases), tanh is the hyperbolic tangent activation function and $\mathbf{W}$, $\mathbf{U}$, $\mathbf{V}$ are coefficient matrices. Note that to simplify the notation we have omitted the layer index in (11): to be specific, $\mathbf{h}_t$ really represents $\mathbf{h}_t^{(H)}$ (the ultimate hidden layer), $\mathbf{h}_{-1}$ stands for $\mathbf{h}_t^{(H-1)}$ (the penultimate layer). Finally, at each step t the first layer $\mathbf{h}_t^{(0)}$ of the forward propagation corresponds to $\mathbf{y}_{t-1:t-p}$.

A shortfall of generics recurrent neural networks is that they fail to capture long-term dependencies. More specifically, if parameters are propagated over too many stages in a simple RNN it typically suffers from the problem of **vanishing gradients** (Goodfellow, Bengio, and Courville 2016). Fortunately, there exist effective extensions of the RNN, most notably the long short-term memory (LSTM), which is our model of choice for Deep VARs. The key idea underlying LSTMs is to regulate exactly how much information is propagated from one cell state vector $\mathbf{C_{t-1}}$ to the next $\mathbf{C_t}$ through the introduction of so called sigmoid gates:

> "The LSTM [has] the ability to remove or add information to the cell state, carefully regulated by structures called gates. Gates are a way to optionally let information through." — (**???**)

These regulating gate layers include a **forget gate** $\mathbf{f}_t$, an **input gate** $\mathbf{i}_t$ and a **output gate** $\mathbf{o}_t$. Each of them are vector-values sigmoid functions whose elements $\mathbf{f_{it}}, \mathbf{i_{it}}, \mathbf{o_{it}}$ are bound between 0 and 1. Their individual purposes are implied by their names: faced with $\mathbf{h_{t-1}}$ and $\mathbf{y}_{t-1:t-p}$, the forget gate regulates how much of each individual unit in $\mathbf{C_{t-1}}$ is retained. Then the input gate regulates which units of $\mathbf{C_{t-1}}$ should be updated and to what candidate values $\tilde{\mathbf{C}}_{\mathbf{t-1}}$. Using the previous two steps the actual update is performed according to the following rule

$$\mathbf{C_t} = \mathbf{f_t} \odot \mathbf{C_{t-1}} + \mathbf{i_t} \odot \tilde{\mathbf{C}}_{\mathbf{t-1}} \tag{12}$$

where $\odot$ indicates the element-wise product. Finally, the output gate acts like a filter on $\mathbf{C_t}$: the new hidden state is computed as $\mathbf{h_t} = \mathbf{o_t} \odot \tanh(\mathbf{C_t})$ where as before we use the hyperbolic tangent as our activation function.[1] Formally, we can summarize the LSTM neural network underlying our Deep VAR framework as follows

$$\begin{aligned}
\mathbf{f}_t &= \sigma\left(\mathbf{b}_f + \mathbf{W}_f \mathbf{h_{t-1}} + \mathbf{U_f} \mathbf{h_{-1}}\right) \\
\mathbf{i}_t &= \sigma\left(\mathbf{b}_i + \mathbf{W}_i \mathbf{h_{t-1}} + \mathbf{U_i} \mathbf{h_{-1}}\right) \\
\mathbf{o}_t &= \sigma\left(\mathbf{b}_o + \mathbf{W}_o \mathbf{h_{t-1}} + \mathbf{U_o} \mathbf{h_{-1}}\right) \\
\mathbf{C_t} &= \mathbf{f}_t \odot \mathbf{C}_{t-1} + \mathbf{i}_t \odot \tanh\left(\mathbf{b}_C + \mathbf{W}_C \mathbf{h_{t-1}} + \mathbf{U_C} \mathbf{h_{-1}}\right) \\
\mathbf{h_t} &= \mathbf{o_t} \odot \tanh(\mathbf{C_t}) \\
\hat{\mathbf{y}}_\mathbf{t} &= \mathbf{c} + \mathbf{V} \mathbf{h_t}
\end{aligned} \tag{13}$$

which is best understood when read from top to bottom. Once again we have simplified the notation we have omitted the layer index in (11). The same notation as before applies.

5.4 Model selection

There are at least two important modelling choices to be made in the context of conventional VARs. The first choice concerns properties of the time series data itself, in particular the order of integration and cointegration. The second choice is about the the lag order p. In order to arrive at appropriate decisions regarding these choices the VAR literature provides a set of guiding principles. We propose to apply these same principles to the Deep VAR, firstly because they are intuitive and simple and secondly because treating both models equally to begin with allows for a better comparison of the two models at the subsequent modelling stages.

5.4.1 Stationarity

When working with time series we are generally concerned about stationarity. Broadly speaking stationarity ensures that the future is like the past and hence any predictions we make based on

[1] For a clear and detailed exposition see (**???**).

past data adequately describe future outcomes. In order to state stationarity conditions in the VAR context it is convenient to restate the K-dimensional VAR(p) process in companion form as

$$\mathbf{Y}_t = \begin{pmatrix} \mathbf{c} \\ 0 \\ \vdots \\ 0 \end{pmatrix} + \mathbf{A}\mathbf{Y}_{t-1} + \begin{pmatrix} \mathbf{u}_t \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{14}$$

where $\mathbf{Y}_t = (\mathbf{y}_t^T, ..., \mathbf{y}_{t-p+1}^T)^T$ and $\mathbf{A}$ is referred to as the companion matrix (Kilian and Lütkepohl 2017). Stationarity of the VAR(p) follows from stability: a VAR(p) is stable if the effects of shocks to the system eventually die out. Stability can assessed through the system's autoregressive roots or equivalently by looking at the eigenvalues of companion matrix $\mathbf{A}$ (Kilian and Lütkepohl 2017). In particular, for the VAR(p) in (14) to be stable we condition that the Kp eigenvalues λ that satisfy

$$\det(\mathbf{A} - \lambda \mathbf{I}_{Kp}) = 0$$

are all of absolute value less than one. Stability implies that the first and second moments of the VAR(p) process are time-invariant, hence ensuring weak stationarity (Kilian and Lütkepohl 2017).

A straight-forward way to deal with stationarity of VARs is to simply ensure that the individual time series entering the system are stationary. This usually involves differencing the time series until they are stationary: for any time series y_i that is integrated of order $I(\delta)$, there exists a δ-order difference that is stationary. An immediate drawback of this approach is the loss of information contained in the levels of the time series. Modelling approaches that take into account conintegration of individual time series can ensure system stationarity and still let individually non-stationary time series enter the system in levels (Hamilton 1994).

5.4.2 Lag order

The VARs lag order p can to some extent be thought of as the persistency of the process: past innovations that still affect outcomes in time t happened at most p periods ago. From a pure model selection perspective we can also think of additional lags in terms of additional regressors that add to the model's complexity. From that perspective choosing a lower lag order corresponds to a form a regularization as it pertains a more parsimonious model.

Various strategies have been proposed to estimate the true or optimal lag order p empirically (Kilian and Lütkepohl 2017). Among the most common ones are sequential testing procedures and selection based on information criteria. The former involves sequentially adding or removing lags - **bottom-up** and **top-down** testing, respectively - and then testing model outcomes in each iteration. A common point of criticism of sequential procedures is that the order tests matters.

Insert reference to Lütkepohl (2005, Section 4.2.3)

Here we will focus on selection based on information criteria, which to some extent makes the trade-off between bias an various explicit (Kilian and Lütkepohl 2017). In particular, it generally involves minimizing information criteria of the following form

$$C(m) = \log(\det(\hat{\Sigma}(m))) + \ell(m) \tag{15}$$

where $\hat{\Sigma}$ is just the sample estimate of the covariance matrix or errors and ℓ is a loss function that penalizes high lag orders. In particular, we have that our best estimate of the optimal lag order p is simply

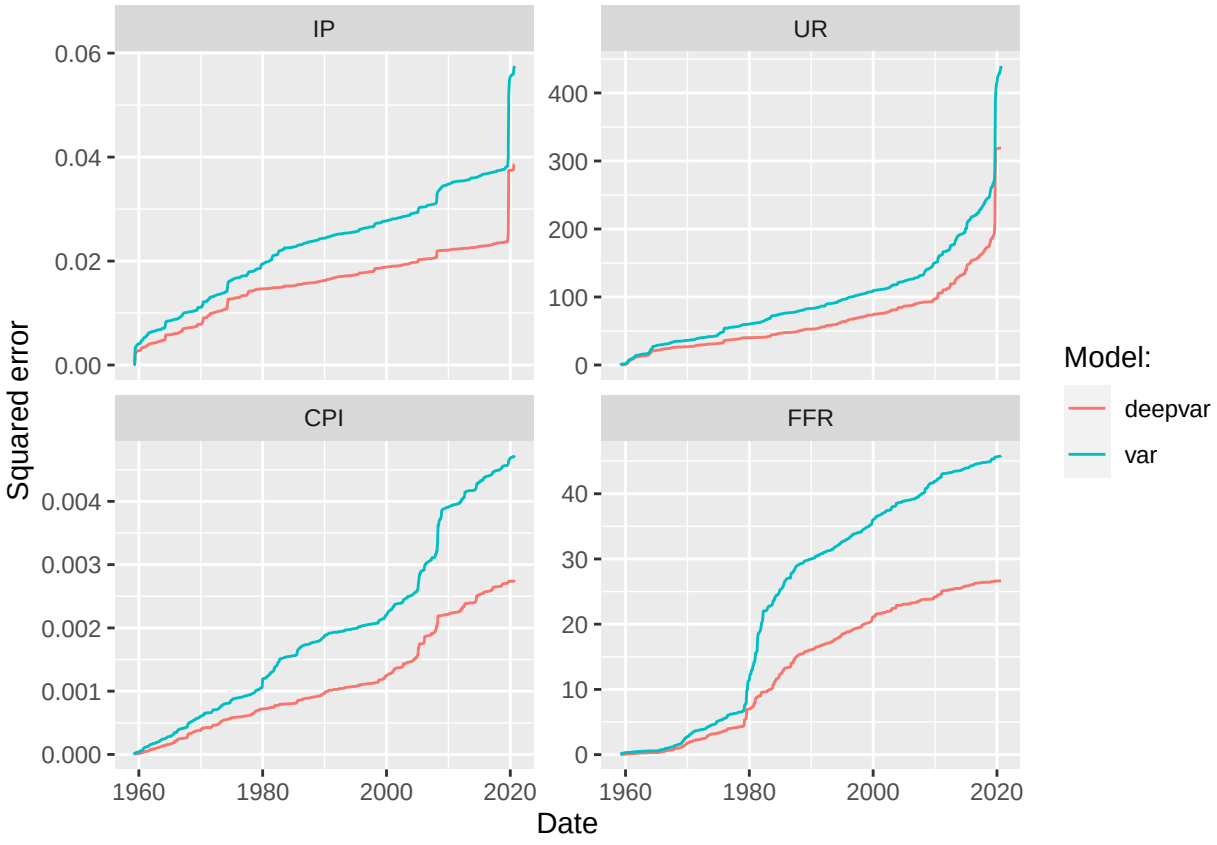
$$\hat{p} = \arg\min_{m \in \mathcal{P}} C(m) \tag{16}$$

where $\mathcal{P} = [m_{\min}, m_{\max}]$. We will consider all of the most common functional choices for (15).

5.4.3 Neural Network Architecture

...

6 Empirical results



In this section we present the main empirical results. First of all, we compare the in-sample fit of the VAR and the Deep-VAR. Then, we proceed with the out-of-sample predicitions. In both cases we compute the cumulative loss for the two methods. Finally, we use the fitted models to produce forecasts.

In figure XX we can see the cumulative RMSE of both the VAR model and Deep-VAR model for each of the time series. The first thing we can observe is that the RMSE of the Deep-VAR is consistently flatter than the RMSE of the VAR. Indicating that the the performance of the Deep-VAR model is better than the one of the VAR throughout the time period of the experimental analysis. This simple fact implies that there are non linear relationships present in the time series. Another interesting insight we can derive form the figure is that the RMSE of the Deep-VAR ends up being even less than half of the RMSE of the VAR, clearly indicating that, overall, the Deep-VAR model outperforms the VAR model for the in-sample fit.

Figure XX is especially useful to asses in which specific periods the Deep-VAR model accomplishes much better fitting results than the VAR model. From the very beginning, we can observe that the slope of the VAR model is grater than the one of the Deep-VAR model, indicating that even in normal economic times, soft non linear relationships may be present in the series.

The first change we can observe is in the beginning of the 1980s. During the beginning of this decade, the RMSE of the Deep-VAR almost remains constant (it slope is close to zero), therefore, it almost makes no error. On the other hand, the RMSE of the VAR keeps increasing. For the case of the IP and the UR, the RMSE of the VAR maintains its previous slope, meaning no improvement nor deterioration appears. Whereas for the CPI, it becomes approximately as twice as big, hence doubling its error, and for the FFR it is approximately quadrupled.

This can be explained by the early 1980s recession. This recession is considered to be the most sever since World War II. As figure XX shows, the Deep-VAR model almost perfectly fits the data in this period. This is due to the fact that economic recessions have a very high component of non linear relationships, therefore not only the Deep-VAR does better than the VAR, but it almost accomplishes a perfect fit. This idea is then reinforced by the poor performance of the VAR in this period for the CPI and FFR series. Both series, specially the FFR, are very volatile in this period. This means that non linear relationships are present and therefore, the Deep-VAR model remains unaffected by the volatility, while the VAR model struggles a lot to fit the data due to its limitations for fitting data generating processes with non linear components.

For the case of the 2007 recession we can see a pattern similar to the one for the recession of the early 1980: the Deep-VAR achieves a better fit because it can correctly identify the pattern soon enough. Furthermore, in the recovery time, the Deep-VAR cumulative RMSE still has a lower slope than the VAR cumulative RMSE.

Finally, we can compare how both models perform in a period with highly non linear components, the Covid era. It can be clearly seen that, for both IP and UR, that is, the series that really changed its behavior with Covid, the VAR model struggles a lot to fit the data. For the VAR model, both series add in a few months approximately 50% of the error tehy had been accumulating for the previous 60 years. Whereas in the case of the Deep-VAR model, they only accumulate approximately 30% of the error they had been accumulating before. This, once again, reinforces our hypothesis that the Deep-VAR model structure helps dealing with non linarities on the data generating process and therefore fitting the data better on periods where non-linareities are present, which means that the Deep-VAR model can be extremely useful for recession periods.

6.1 Train-test split

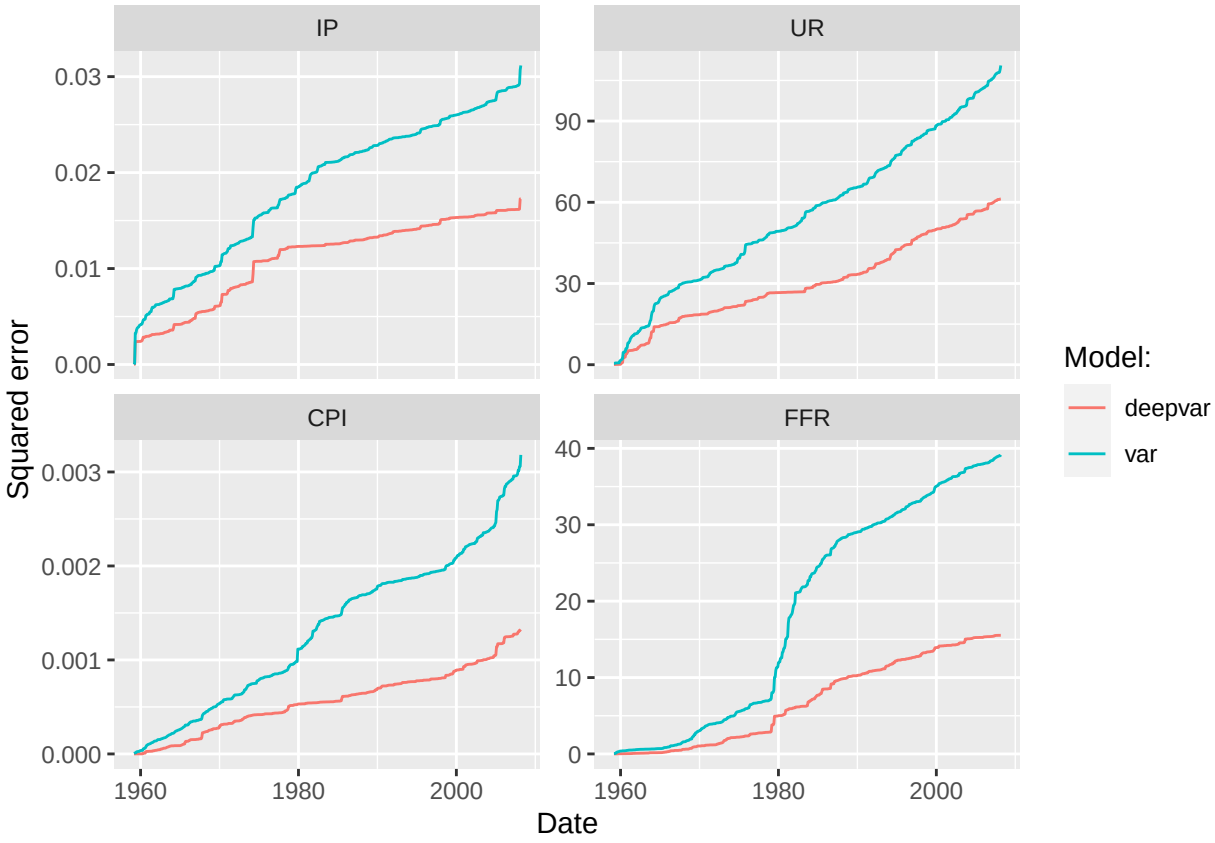
Now, we split the data into training data and test data. The model will be fitted in the training dataset and then we assess the performance of the model in the test dataset. This is done in order

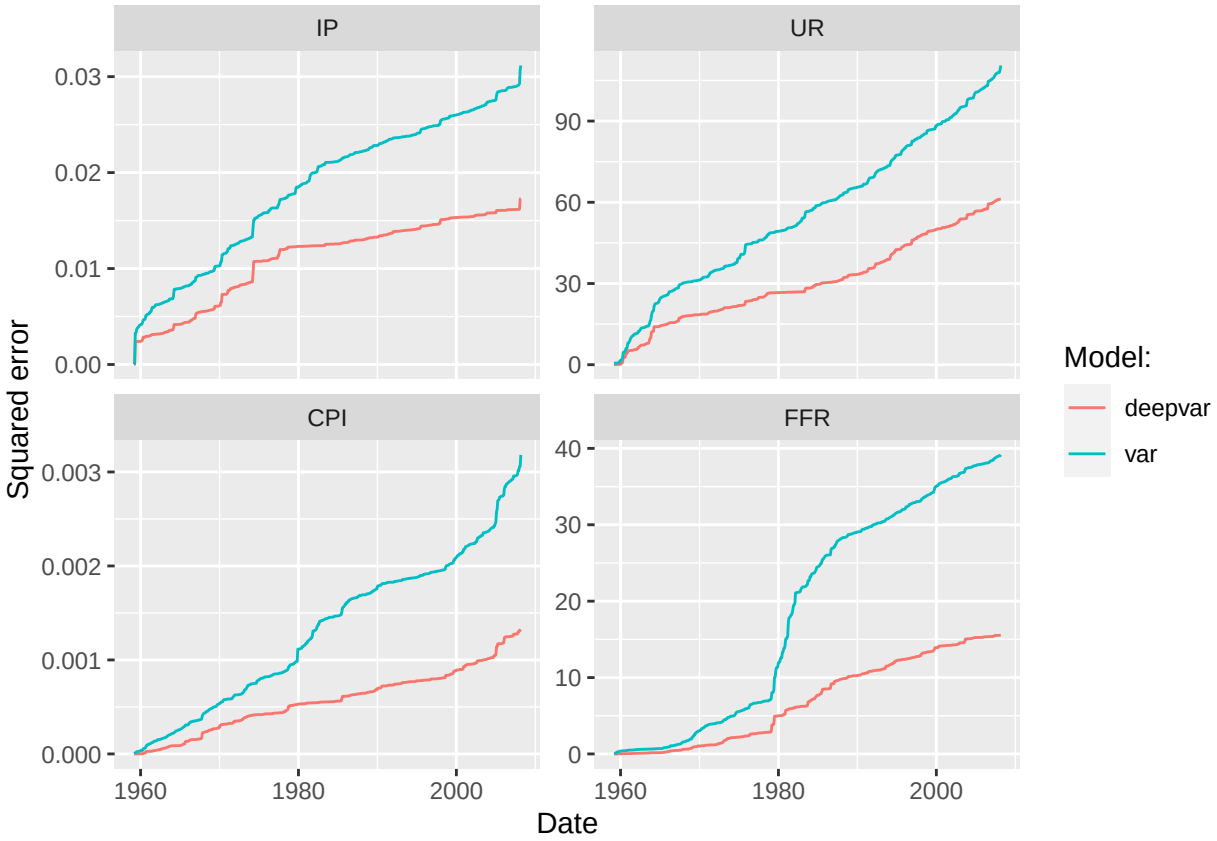
to see the performace of both models on data which is independent of the data that has been used to fit the model. The training dataset goes from March 1959 to November 2007, whereas the test data goes from December 2007 to March 2021, so we train the model on 80 percent of the data and we test the model on the remaining 20 percent.

The following table shows ths Root Mean Squared Error (RMSE) for the in-sample and the out-of-sample predictions of both the VAR model and the Deep-VAR model. We can see that the RMSE for the Deep VAR outperforms the one for the classic VAR for both the training data and the test data and for all time series. The fifth column of the table shows us the ratio between the Deep-VAR and the VAR. The lower the ratio, the better the Deep-VAR with respect to the VAR. Note that for the industrial production and for the unemployment rate the Deep-VAR does fairly well compared to the VAR.

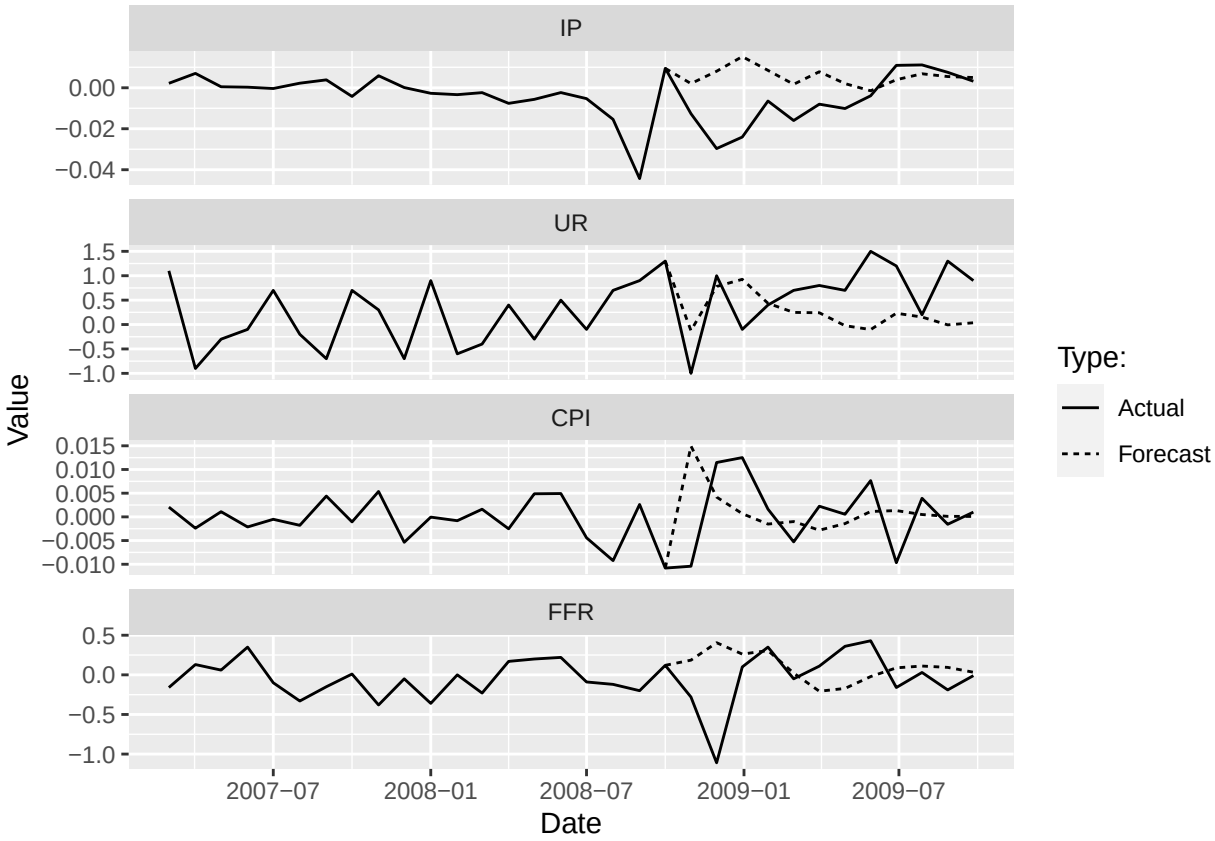
Sample	**Variable**	**DeepVAR**	**VAR**	**Ratio**
test	IP	0.01	0.01	0.40
test	UR	0.98	1.65	0.59
test	CPI	0.00	0.00	0.79
test	FFR	0.21	0.24	0.86
train	IP	0.01	0.01	0.74
train	UR	0.32	0.43	0.74
train	CPI	0.00	0.00	0.64
train	FFR	0.16	0.26	0.63

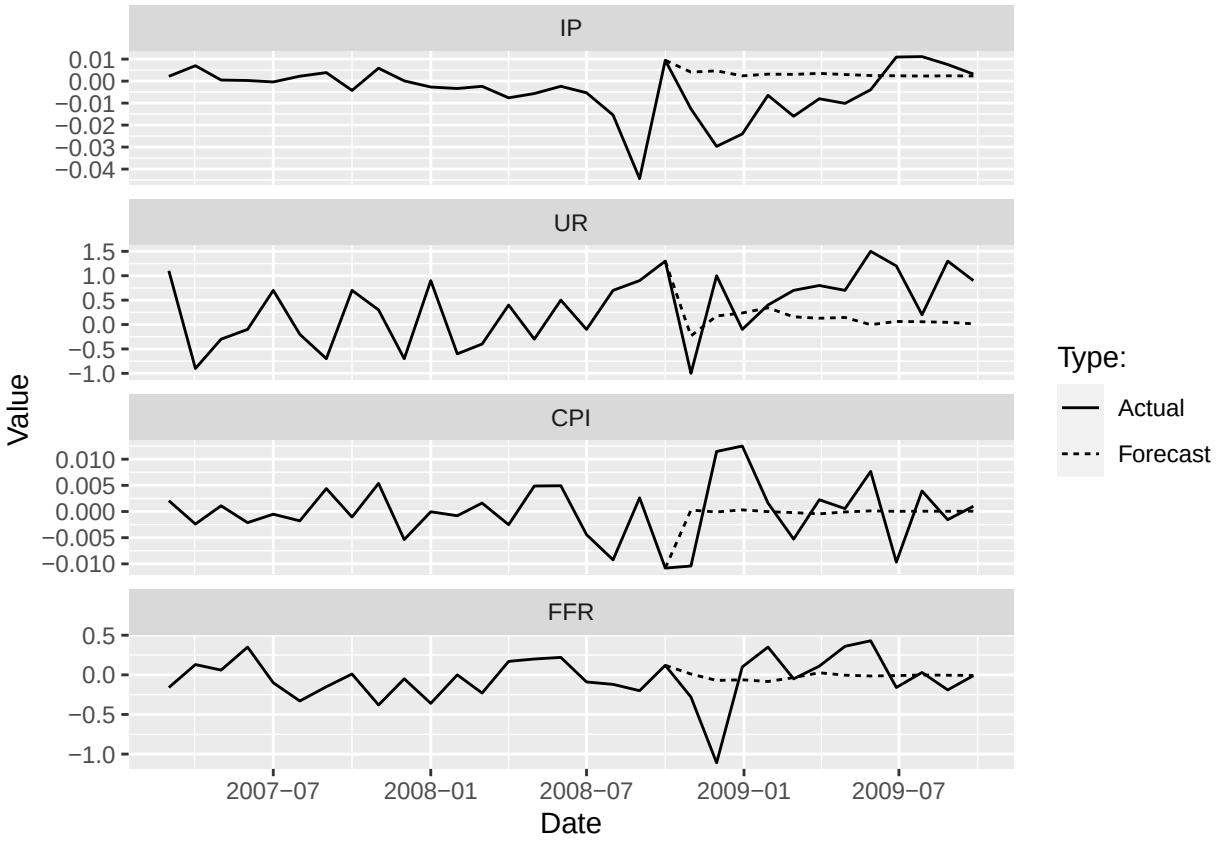
Table 4: RMS


IP
UR
CPI
FFR
Squared error
Date
Model:
deepvar
var
0.03
0.02
0.01
0.00
90
60
30
0
0.003
0.002
0.001
0.000
40
30
20
10
0
1960
1980
2000



In the following, we will compare the forecast performance of the VAR and the Deep-VAR. To do that, we compute the 1-step ahead forecast for one year. The dashed line represents the forecast and the solid black line represents the actual value of the time series. To compare the performance of both methods, we compute the Forecast RMSE.


IP
UR
CPI
FFR
Value
Date
2007−07
2008−01
2008−07
2009−01
2009−07
Type:
Actual
Forecast



In the following table, we look more specifically at this comparison. In particular, we compute the Forecast RMSE for the VAR method and for the Deep-VAR, and we also compute the correlation of the forecast with the actual value of each time series.

Variable	VAR FRMSE	Deep-VAR FRMSE	VAR correlations	Deep-VAR correlations
IP	0.02	0.02	-0.30	-0.67
UR	0.39	0.16	-0.10	0.26
CPI	0.01	0.01	-0.34	0.05
FFR	0.20	0.04	-0.56	0.20

Table 5: Forecasts results

As we can see in the table, the Forecast RMSE of the Deep-VAR is lower than the one for the VAR for unemployment rate and for the federal funds rate, and it is the same for industrial production and inflation. Hence, the Deep-VAR outperforms the VAR in terms of forecasts. It is also of interest to analyze the correlation between the forecasts and the actual values. In the case of the VAR, there is, for all time series, a negative correlation, that is, when the time series evolves in one direction, the VAR forecast evolves in the opposite direction. However, this is not true for the Deep-VAR. For industrial production, is certainly true that the Deep-VAR forecast has a highly negative correlation with the actual values, but achives the same forecasting performance than the one for the VAR. However, for the rest of time series, the Deep-VAR has a positive correlation, that is, the forecast generally evolves in the same direction as the actuial values. Note that this

positive correlation cannot be too strong given that the forecasts at some point in time return to the mean, and hence, have no varaiability.

6.2 Grid search

7 Caveats and extensions

In this section we will talk about the main short-falls of our work, and potentital ways to improve it.

In this paper we have developed a method to produce point forecasts which has shown to over-perform the traditional VAR. However, policy makers do not only take decisions based on point estimates, but they also take into account how uncertain they are. That is why, when constructing forecats you also want to infer confidence intervals for your point estimates. Doing that with the VAR methodology is not difficult, given that standard errors are inherent to the model, and easily attainable. This is not the case for the Deep VAR, but this does not mean that we cannot obtain them but that we need to rely on bootstraping or montecarlo techniques to infer these bounds. This relates to the dropout rate used when fitting the Deep VAR in the test set. Given that the dropout rate specifies the number of observations that the model randomly removes at each layer, each time the model is refitted the test set will result in differnt but similar predictions. Threby, this can be used to get the standard errors needed for reporting the confidence intervals. On top of that, if for each simulation we vary the dropout rate, the predictions will range in a wider interval. This approach of refitting the model in the test set many times with differnt dropout rates can be used for getting convidence intervals for the predictions of the test set.

Impulse response functions are another missing milestone in our paper and to which future research should be dedicated. IRFs are important to see how your model captures the dynamics of the variables in response to a shock in a given period of time. When estimating the model with the traditional VAR, computing IRFs is quite straighforward. Provided that all the roots of the autoregressive polynomial lie outside the unit circle, transforming the model into its moving average representation and plugging the shock into the vector moving average (VMA) representaion resutls into the IRF of the shock analysed. However, generating the IRFs is more difficult in the Deep VAR setting given that we cannot get the Deep VMA representation as a consequence of its underlying structure. Thereby, future work would be needed in this direction to be able to derive the IRFs from the Deep VAR model. The idea would be ...

Within the VAR framework, it is quite usual to obtain the Forecast Error Variance Decomposition (FEVD) as well. This allows the researcher or the policy maker to exactly know what percentage of the variability of the forecast error of one variable is explained by the other variables in the system. However, is not clear how to get that from the Deep VAR perspective. Our intuition is that, in the same way you can use montecarlo simulations to get different estimations through the dropput rate and therefore get some variability to infer the standard error, you could also construct different forecasts. These forecasts would lead to different forecast errors and threfore potentially ending up reproducing the FEDV for the Deep VAR.

Apart from the forecasts and the previouly mentioned important insights that can be derived from the VAR model and that could potentially be derived from the Deep VAR model with further work, poilcy makers are also concerned about the interpretabily of the model. The linear relationship

of inputs and outputs of the VAR allows the researcher or policy maker to asses the effect of one variable of the system to another and threfore assessing if that variable granger causes the other. In the case of the Deep VAR model, its non linar structure makes it impossible to recover this assiciatinos. There has been some research already devoted to this area. For instance the Neural Aditive Vector Autorregression (NAVAR) aims to combine neural networks in the first stage of the model use a linear additive structure in the end in order to assess if one varible granger causes another. It uses a NN for each variable in the system to output the degree of causality of this variable to all the variables in the system in the form of contributions. Then, once the model has assessed the contribution of all the variables to themself and the others, an additive structure is implemented to assess the contributions of the system to each of the variables.

References

Bernanke, Ben. 1990. "The Federal Funds Rate and the Channels of Monetary Transnission." National Bureau of Economic Research.

Brock, William A, David Arthur Hsieh, Blake Dean LeBaron, William E Brock, and others. 1991. *Nonlinear Dynamics, Chaos, and Instability: Statistical Theory and Economic Evidence*. MIT press.

Cortes, Corinna, and Vladimir Vapnik. 1995. "Support-Vector Networks." *Machine Learning* 20 (3). Springer: 273–97.

Dorffner, Georg. 1996. "Neural Networks for Time Series Processing." In *Neural Network World*. Citeseer.

Enders, Walter. 2008. *Applied Econometric Time Series*. John Wiley & Sons.

Fix, E, and J Hodges. 1951. "An Important Contribution to Nonparametric Discriminant Analysis and Density Estimation." *International Statistical Review* 3 (57): 233–38.

Friedman, Milton, and Anna Jacobson Schwartz. 2008. *A Monetary History of the United States, 1867-1960*. Princeton University Press.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.

Greene, William H. 2012. "Econometric Analysis 7th Ed (International)." Pearson.

Hamilton, James Douglas. 1994. *Time Series Analysis*. Princeton university press.

Hamzaçebi, Coşkun. 2008. "Improving Artificial Neural Networks' Performance in Seasonal Time Series Forecasting." *Information Sciences* 178 (23). Elsevier: 4550–9.

Ho, Tin Kam. 1995. "Random Decision Forests." In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, 1:278–82. IEEE.

Hochreiter, Sepp, and Jürgen Schmidhuber. 1997. "Long Short-Term Memory." *Neural Computation* 9 (8). MIT Press: 1735–80.

Joseph, Andreas, Eleni Kalamara, George Kapetanios, and Galina Potjagailo. 2021. "Forecasting Uk Inflation Bottom up."

Kihoro, J, RO Otieno, and C Wafula. 2004. "Seasonal Time Series Forecasting: A Comparative Study of Arima and Ann Models." African Journal of Science; Technology (AJST).

Kilian, Lutz, and Helmut Lütkepohl. 2017. *Structural Vector Autoregressive Analysis*. Cambridge University Press.

Kydland, Finn E, and Edward C Prescott. 1982. "Time to Build and Aggregate Fluctuations." *Econometrica: Journal of the Econometric Society*. JSTOR, 1345–70.

Lucas Jr, Robert E. 1976. "Econometric Policy Evaluation: A Critique." In *Carnegie-Rochester Conference Series on Public Policy*, 1:19–46. North-Holland.

McCulloch, Warren S, and Walter Pitts. 1990. "A Logical Calculus of the Ideas Immanent in Nervous Activity." *Bulletin of Mathematical Biology* 52 (1-2). Springer: 99–115.

Romer, Christina D, and David H Romer. 1989. "Does Monetary Policy Matter? A New Test in the Spirit of Friedman and Schwartz." *NBER Macroeconomics Annual* 4. MIT Press: 121–70.

Sims, Christopher A, and others. 1986. "Are Forecasting Models Usable for Policy Analysis?" *Quarterly Review* 10 (Win). Federal Reserve Bank of Minneapolis: 2–16.

Verstyuk, Sergiy. 2020. "Modeling Multivariate Time Series in Economics: From Auto-Regressions to Recurrent Neural Networks." *Available at SSRN 3589337*.

Zhang, G Peter. 2003. "Time Series Forecasting Using a Hybrid Arima and Neural Network Model." *Neurocomputing* 50. Elsevier: 159–75.

Zhang, Guoqiang, B Eddy Patuwo, and Michael Y Hu. 1998. "Forecasting with Artificial Neural Networks:: The State of the Art." *International Journal of Forecasting* 14 (1). Elsevier: 35–62.

A R Code and Package

All code used for the empirical analysis presented in this article can be found on the corresponding GitHub repository. Researchers interested in using Deep VARs more generally for their own empirical work may find the R `deepvars` package useful which is being maintained by one of the authors. The package is still under development and as of now only available on GitHub. To install the package in R simply run:

```
devtools::install_github("pat-alt/deepvars", build_vignettes=TRUE)
```

Package vignettes will take you through the basic package functionality. Once the package has been installed simply run `utils::browseVignettes()` to access the documentation.