Modeling Multivariate Time Series in Economics: from Auto-Regressions to Recurrent Neural Networks*

Sergiy Verstyuk†

This draft: 7 December 2018
First draft: 30 November 2018

Abstract

The modeling of multivariate time series in an agnostic manner, without assumptions about underlying theoretical structure is traditionally conducted using Vector Auto-Regressions. They are well suited for linear and state-independent evolution. A more general methodology of Multivariate Recurrent Neural Networks allows to capture non-linear and state-dependent dynamics. This paper takes a range of small- to large-scale Long Short-Term Memory MRNNs and pits them against VARs in an application to US data on GDP growth, inflation, commodity prices, Fed Funds rate and bank reserves. Even in a small-sample regime, MRNN outperforms VAR in forecasting: its out-of-sample predictions are about 20% more accurate. MRNN also fares better in interpretability by means of impulse response functions: for instance, a temporary shock to the Fed Funds rate variable generates system dynamics that are more plausible according to conventional economic theory.

*Acknowledgements: Jörn Boehnke, Scott Kominers, Jun Liu; as well as the staff of FAS Research Computing at Harvard University.

†Contact address: verstyuk@cmsa.fas.harvard.edu.

Contents

1 Introduction

Artificial Neural Networks are mathematical models for pattern extraction and recognition, a task they achieve by passing their inputs through a series of non-linear transformations and mapping them to targeted outputs; hence the term Deep Learning. Over the last several years this machine learning approach demonstrated impressive progress in image, text/sound and video recognition. Success in image processing became possible due to Convolutional Neural Networks that work well for spatial data. Success in text and sound processing was made to a large extent due to Recurrent Neural Networks that work well for sequential data. This paper investigates ANNs applicability to multidimensional time series data that are encountered in economics, naturally focusing on the RNN family, and on its Long Short-Term Memory member more specifically.

Recurrent Neural Network maintains within its network the connections that look back in time. When processing sequential data, this allows RNN to account for a relatively distant history, to keep track of the context, and to handle data of multiple frequency or scales. For example, in a natural language processing application within the context of geography-related words such as "city" or "region" an RNN after encountering the word "French" would predict a high probability of encountering word "Toulouse" next, but within the context of food-related words such as "bread" or "butter" it would predict instead a high probability of "baguette" coming next.

In this paper we are interested in agnostic statistical modeling that does not rely on "structural" restrictions stemming from economic theory. Traditionally in economics this is implemented using Vector Auto-Regressions, a simple time-series model that we use as a benchmark. On the other hand, LSTM Multivariate Recurrent Neural Network fulfills the same role, addtionally allowing for (i) highly non-linear relationships, as well as (ii) state-dependent dynamics in the data. In fact, a VAR model can be viewed as a special case of MRNN.

The data set we use here comprises GDP growth, inflation, commodity prices, Fed Funds rate, and bank reserves for the United States since 1959 (at monthly frequency, after the aggregation/disaggregation procedures described in the text).

In our exercise MRNN architectures of varying depth (from 1 to 4 hidden layers) and width (from 5 to 500 units per layer) are considered. In the case of large-capacity NNs, we allow for substantial redundacy, since available regularization techniques such as dropout prevent overfitting quite effectively. MRNN is trained using mini-batch gradient descent. VAR is estimated by OLS, trying various lag lengths. In both cases, the process of parameter learning is guided by prediction performance on the validation sub-sample.

Basically, we run a genuine horse-race between VAR and MRNN, providing two models with the same data sets and letting them use the data in the most effective way their structure allows. Then we compare the fruitfulness of the two modeling approaches.

Our preferred LSTM MRNN architecture is fairly minimalistic, comprising 2 hidden layers with 10 units in each. VAR specification we ended up with comprises 4 lags.

This paper's approach is rooted in modern machine learning that focuses on predicting data out-of-sample, as opposed to more traditional econometrics that deals with fitting data in-sample. Because of this, and also for the reasons of practical interest, the first, quantitative domain for comparing two modeling methods is prediction performance. Our LSTM MRNN outperforms VAR by about 20% in prediction accuracy on the testing subsample (i.e., in generalization).

The second, qualitative domain is related to an aspect of model interpetation that is revealed by impulse responses. These are a standard exercise for VARs, and in this paper we propose a simple procedure of their implemention for MRNNs. We examine the commonly considered situation of a temporary increase in the Fed Funds rate, and verify which of the two models behaves more in line with economic theory.

It turns out that following such an impulse, the models perform more or less similarly, except in a single yet important aspect. VAR generates a prolonged increase of inflation rate in the wake of monetary contraction. This is a theoretically controversial empirical finding that is referred to as a "price puzzle" in the literature (Sims, 1992; Uhlig, 2005), although some authors do offer so-called "Neo-Fisherian" explanations behind it (Cochrane, 2018). MRNN, on the other hand, produces theoretically more plausible response dynamics: a small increase of inflation in the short run, but a subsequent much larger decline in the long run. Putting this into context, in order to exhibit response dynamics that are more in line with economic intuition, VARs usually require an introduction of additional theoretical ("structural") identification restrictions. But MRNN, at least in this particular exercise, appears to identify them from raw data.

Lastly, it is worth noting that we have a fairly small data set: just 59 years of monthly-frequency data on 5 economic variables. The parameterizations of our two models are quite rich: VAR comprises 105, while MRNN comprises 1535 parameters. The conventional views are skeptical about the viability of statistical models in such situations, especially with regard to "data-hungry" NN architectures. However, there are theoretical arguments why NNs effective capacity is in practice substantially lower than what a raw parameter count suggests (see the main text); this point is indirectly supported by strong performance of our MRNN in out-of-sample predictions. Also in our defense, we did attempt a data augmentation exercise, as one way to synthetically expand the limited data set; it did not seem to bring any benefit though.

2 Literature

Artificial Neural Networks and Deep Learning is a very rapidly advancing area of knowledge. Excellent textbook treatment is available in a still relevant Bishop (2006), as well as in a less formal but more up-to-date Goodfellow et al. (2016). A very broad review is available in Schmidhuber (2015), a good review with an eye on applications to Natural Language Processing is Goldberg (2016), a review of Recurrent Neural Networks if offered

in Lipton et al. (2015).

Important ANN architectures include Feed Forward Neural Networks that were historically first (single-layer due to Rosenblatt, 1958; multi-layer proposed by Ivakhnenko and Lapa, 1965), Convolutional Neural Networks used for image recognition (LeNet introduced by LeCun et al., 1998; AlexNet by Krizhevsky et al., 2012; GoogLeNet by Szegedy et al., 2014), Recurrent Neural Networks used for text/sound processing (Elman, 1990). Here we focus on the latter. (Interestingly, according to Liao and Poggio (2016), RNNs are biologically-plausible models of the cortex—thus closing the loop back to seminal McCulloch and Pitts (1943), whose pioneering NN ideas lied on the interface of biology and computer science.)

Recurrent Neural Network forms a network that loops to itself, which allows to account for recent history when processing sequential data. Specifically, we focus on Long Short-Term Memory variation of RNN, which has the facilities to prevent the problem of vanishing/exploding error gradients used in the Back-Propagation Through Time algorithm (Hochreiter and Schmidhuber, 1997).

Apart from theoretical existence results regarding accurate approximation (Cybenko, 1989; Hornik et al., 1989; also see Kolmogorov, 1957; and Arnold, 1957), there is still no consensus about the reasons for NN effectiveness. Possible explanations come from statistical physics (Mehta and Schwab, 2014), geometry (Lei et al., 2018), information theory (Schwartz-Ziv and Tishby, 2017), Bayesian statistics (Patel et al., 2015; Polson and Sokolov, 2017; Friston et al., 2018). Arguably, an important role is played by clever training and regularization techniques (such as Hinton and Camp, 1993; Hochreiter and Schmidhuber, 1997; Srivastava et al., 2014; Zaremba et al., 2015; Mandt et al., 2017; and many others).

Besides well-known practical applications, ANNs gain wide usage in natural science and mathematics. For instance, RNNs have been successfully applied to high-dimensional chaotic systems (Jaeger and Haas, 2004; Vlachas et al., 2018; Pathak et al., 2018). NNs also found application in cryptography (Maghrebi et al., 2016; Cagli et al., 2017), and even to data from geometry and physics that is relevant to string theory (He, 2017).

Economics is also not lagging behind. ANNs have been used for estimation of causal relationships developing the broad idea of Instrumental Variables (Hartford et al., 2016), for auction design (Dütting et al., 2017), for finance (Heaton et al., 2016a; 2016b; Sirignano, 2016; Feng et al., 2018; Gu et al., 2018).

Considering applications to sequential data, a good source is Graves and Schmidhuber (2009), who use multidimensional LSTM RNN for handwriting recognition. A more recent paper is Goel et al. (2016), with their application of LSTM to multivariate time series from aviation industry. One application to sequential data from the field of finance is Dixon et al. (2018), although instead of RNN the authors use a simpler FFNN with lagged explanatory variables embedded into an input vector.

As a benchmark for comparison, we rely on Vector Auto-Regressions. Loosely speak-

ing, they combine Auto-Regressive Integrated Moving Average models (Box and Jenkins, 1970) and models comprising systems of equations (Theil, 1954; Zellner, 1962; Zellner and Theil, 1962). In economics, this approach was popularized by Sims (1980). Thanks to their simplicity, VARs are still extensively used. A recent review of VAR and time series analysis more generally is available in Stock and Watson (2017), a review of the so-called structural VARs is offered by Stock and Watson (2016).

In this paper we apply MRNN and VAR models to macroeconomic data from the United States, with an eye on forecasting and policy analysis. Relevant references are provided in the text sections where these applications are discussed in detail.

3 Methodology

3.1 Framework

First, let us clarify some of the terminology used going forward. We divide the whole universe of Artificial Neural Network models into different architectures (or specifications), and we divide each architecture into its different trained instances.

Given an architecture, the process of hyperparameter tuning (a kind of meta-optimization) defines the parameters ruling the optimization process.[1] Then, optimization, or learning, algorithm executes the model's parameter training (as opposed to estimation in more traditional statistical settings). Next, regularization techniques help with the trained model's parameter refinement (note that optimization and regularization are often conducted in an alternating manner). Lastly, model selection means: (a) for a specific architecture, the choice among trained instances of the best one; and/or (b) between different architectures, the selection of the best one.

3.1.1 Architecture

Artificial Neural Networks, or just nets, are inspired by biological neural networks: units (nodes, cells or "neurons") have connections (edges or "axons", "synapses" and "dendrites") to other units, and they perform complex non-linear computations by transmitting signals from one to another. Mathematically, ANN is a model defined by the following system of equations (or architecture):

$$\begin{aligned}
\boldsymbol{h}_1 &= \boldsymbol{g}_1\left(\boldsymbol{b}_1 + \boldsymbol{W}_1\boldsymbol{x}\right), \\
\boldsymbol{h}_2 &= \boldsymbol{g}_2\left(\boldsymbol{b}_2 + \boldsymbol{W}_2\boldsymbol{h}_1\right), \\
&\dots \\
\boldsymbol{h}_M &= \boldsymbol{g}_M\left(\boldsymbol{b}_M + \boldsymbol{W}_M\boldsymbol{h}_{M-1}\right), \\
\hat{\boldsymbol{y}} &= \boldsymbol{g}_{M+1}\left(\boldsymbol{b}_{M+1} + \boldsymbol{W}_{M+1}\boldsymbol{h}_M\right),
\end{aligned} \tag{1}$$

[1] Although sometimes hyperparameters are understood as also including the details of the architecture itself.

where $\boldsymbol{x} \in \mathbb{R}^{K_x}$ is an a K_x-dimensional input vector; $\boldsymbol{b}_m \in \mathbb{R}^{n_m}$ and $\boldsymbol{W}_m \in \mathbb{R}^{n_m \times n_{m-1}}$ are parameters (biases and weights, respectively); $\boldsymbol{g}_m(\cdot)$ are activation functions such that $\boldsymbol{g}_m : \mathbb{R}^{n_m} \mapsto \mathbb{R}^{n_m}$; $\boldsymbol{h}_m(\cdot)$ are outputs of hidden layers; with hidden layers $m \in \{1, \ldots, M\}$ and hidden units $n_m \in \mathbb{N}^+$ (keeping $n_0 := K_x$); as well as $\hat{\boldsymbol{y}} \in \mathbb{R}^{K_y}$ is the output vector aiming to predict the value of a random variable $\boldsymbol{y} \in \mathbb{R}^{K_y}$, with $\boldsymbol{\xi}$ being the prediction error,

$$\boldsymbol{y} := \hat{\boldsymbol{y}} + \boldsymbol{\xi}. \tag{2}$$

In this paper, $K_x = K_y =: K \in \mathbb{N}^+$.

The above system can be more compactly expressed as a composition of functions

$$\hat{\boldsymbol{y}} = g_M^B \circ \cdots \circ g_1^B(\boldsymbol{z}), \tag{3}$$

where, for $\boldsymbol{B}_m := [\boldsymbol{b}_m, \boldsymbol{W}_m]$ and $\boldsymbol{z} := [1; \boldsymbol{x}]$, we used the notation

$$\boldsymbol{g}_m^B(\boldsymbol{z}) := \boldsymbol{g}_m\left(\boldsymbol{B}_m \boldsymbol{z}\right). \tag{4}$$

Basically, the network passes the incoming data $\boldsymbol{x}$ through several stages of non-linear (semi-linear) transformations, and aims to predict some target $\boldsymbol{y}$. One can think of $\boldsymbol{g}_m(\cdot)$ as units, and of $\boldsymbol{W}_m$ as connections.

For the time-series setup, we are using a particular type of ANN that is known as (Multivariate) Recurrent Neural Network, in particular the specification called Long Short-Term Memory.[2] It specializes the general model along the following lines. First, random variables as well as state variables are time-indexed. Second, the input variable is just a lagged variable that is being predicted by the output, $\boldsymbol{x}_t := \boldsymbol{y}_{t-1}$. Third, $\boldsymbol{g}_{m,t}(\cdot)$ takes a special form (for readability, omitting the layers' subscripts m throughout):

$$\begin{aligned}
\boldsymbol{h}_t &= \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t), \\
\boldsymbol{c}_t &= \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \tanh(\boldsymbol{b}_c + \boldsymbol{W}_c \boldsymbol{h}_{-1,t} + \boldsymbol{U}_c \boldsymbol{h}_{t-1}), \\
\boldsymbol{o}_t &= \sigma(\boldsymbol{b}_o + \boldsymbol{W}_o \boldsymbol{h}_{-1,t} + \boldsymbol{U}_o \boldsymbol{h}_{t-1}), \\
\boldsymbol{i}_t &= \sigma(\boldsymbol{b}_i + \boldsymbol{W}_i \boldsymbol{h}_{-1,t} + \boldsymbol{U}_i \boldsymbol{h}_{t-1}), \\
\boldsymbol{f}_t &= \sigma(\boldsymbol{b}_f + \boldsymbol{W}_f \boldsymbol{h}_{-1,t} + \boldsymbol{U}_f \boldsymbol{h}_{t-1});
\end{aligned} \tag{5}$$

where $\boldsymbol{h}_t \in \mathbb{R}^n$ is in this specification interpreted as a cell hidden state vector (more precisely, due to the readability caveat above, $\boldsymbol{h}_t$ stands for $\boldsymbol{h}_{m,t}$, and $\boldsymbol{h}_{-1,t}$ denotes $\boldsymbol{h}_{m-1,t}$, while $\boldsymbol{h}_{0,t} := \boldsymbol{x}_t$); $\boldsymbol{c}_t \in \mathbb{R}^n$ is a cell memory state vector; $\boldsymbol{i}_t \in \mathbb{R}^n$, $\boldsymbol{f}_t \in \mathbb{R}^n$ and $\boldsymbol{o}_t \in \mathbb{R}^n$ are an input gate, forget gate and output gate activation vectors, respectively; and with $\boldsymbol{b}_\cdot \in \mathbb{R}^n$, $\boldsymbol{W}_\cdot \in \mathbb{R}^{n \times n_{-1}}$, and $\boldsymbol{U}_\cdot \in \mathbb{R}^{n \times n}$ being parameters.

Introduction of state vectors allows the time-series models to condition on recent historical dynamics. It may be helpful to think of LSTM network as a kind of non-linear state-space model (that are very easy to work with due to Kalman-Bucy filter as described, for example, in Hamilton, 1994).

[2] In the specification's title, "long memory" stands for slowly-changing parameters of the ANN, "short memory" for recurrent inputs to the hidden layer from its past outputs, and "long short-term memory" for the amendment of recurrent connections with additional parametric network structures.

3.1.2 Mechanics

An important theoretical result about NNs is that their formulation is general enough to represent any desired function. Given at least one hidden layer and a sufficiently large but finite number of units, this is feasible by the arguments of the universal approximation theorem (Cybenko, 1989; Hornik et al., 1989; also see the representation theorem due to Kolmogorov, 1957; and Arnold, 1957).[3] A useful implication is NNs' ability to account for non-linearity of the empirical phenomena being modeled.

Size (or capacity, parameterization) of a NN is determined by its depth, that is the number of hidden layers M, and its width, that is the number of units in each layer $\{n_1, \ldots, n_M\}$. NNs are called deep if they have a large number of hidden layers.

The above theory requires a large enough NN, but it does not provide us with the formal method for setting a sufficient size and structuring the architecture (and, more generally, for learning the required parameters).[4] Our guiding principle is given by what is regarded as the scientific method: successful prediction on the new data. A theory or model is good only if it generalizes the existing facts in a way that is useful in predicting yet unobserved phenomena.

From the theory, we know that an insufficiently sized NN will underfit the data: the standard statistical fit measures will demonstrate poor performance, even on the observed sample that is used for training ("in-sample"). At the other extreme, a too large NN may overfit the data: statistical fit measures will demonstrate strong performance on the observed sample that was used for training, but will collapse when the model is tested on the previously unobserved data ("out-of-sample"). In the machine learning literature, we say about the latter case that the NN has memorized the data without learning its high-level structure and thus acquiring the ability to generalize the accumulated knowledge.

Thus, the strategy commonly adopted in the literature is extensive exploration that amounts to experimenting with different model specifications, using as a criterion that guides the exploration the predictive performance of the specifications considered. Usually, the predictive fit is measured on a sample reserved exclusively for such validation purposes (and that is separate from the training and testing samples).

One practical approach is to start with a generously sized network, allowing for redundancy and erring on the side of a too many parameters, and then restricting the excessive depth/width using the regularization methods described below.[5] Another practical rule of thumb is that deep architectures tend to be easier and cheaper to train, hence their

[3] Formally, a function that is Borel measurable and defined over finite-dimensional spaces (e.g., continuous on a compact subset of $\mathbb{R}^n$) can thus be approximated with arbitrary precision.

[4] Technically, the above theoretical result is not constructive, it is only about the existence.

[5] Existense of effective regularization methods is the reason why overfitting is less problematic than underfitting (this is a little reminiscent of inefficiency versus inconsistency in the practice of estimating linear regression models, hence the prudent preference for including rather too many than too few explanatory variables). Also, allowing for redundancy improves NN reliability and makes it easier to train.

popularity even though in theory having just one hidden layer is sufficient.

Regularization: The aim of regularization is to improve generalization and reduce overfitting of a given model. The leading motive is Occam's razor principle: the heuristic postulate that good scientific theories are parsimonous. Regularization downplays the weight of the training sample in parameter learning by introducing additional information and constraints, which tends to prevent the fitting to noise and memorization at the expense of generalization.

The recent literature provides many effective regularization techniques. Some of the most popular methods are explicitly using the idea that good models should be invariant to "small" perturbations.

Dropout adds noise to the NN architecture (Srivastava et al., 2014; also see Zaremba et al., 2015). During the training process, it randomly drops from the model some units and connections (usually in the hidden layer, but possibly in the visible one too). This is a very simple yet highly effective regularization method. It can be loosely motivated as mimicking the model averaging (as the authors of Srivastava et al., 2014, orignially suggested) as well as estimation under LASSO/sparsity or Ridge constraint (Wager et al., 2013).

Data augmentation adds noise to the NN inputs (the idea can be traced back to Sietsma and Dow, 1991).[6] This expands the data set with synthetic data comprising the originals that are contaminated with some (usually, Gaussian/white) noise. In such case, effectively we are dealing with a kind of background noise. But noise can also be added exploiting the structure of the data at hand. For visual data, it involves image transformations such as translations, rotations, scaling, cropping, color perturbations (LeCun et al., 1998; Simard et al., 2003; Krizhevsky et al., 2012). For audio data, it involves sound transformations such as frequency, tempo and speed perturbations (Jaitly and Hinton, 2013; Cui et al., 2015; Ko et al., 2015).

Optimization and tuning: NN models are very costly to work with: usually they are truly high-dimensional and have a large number of parameters; they have non-linear structure and non-convex objective functions; they often involve processing very large datasets. Thus, careful design of the whole computational approach is of paramount importance. This paper leaves out the discussion of the hardware and software components (necessary resources such as Graphics Processing Units and machine learning libraries are now widely available), and focuses instead on the algorithmic component that comprises data handling, minimization algorithms, meta-optimization of the latter, etc.

We briefly present some of the techniques that proved to be useful in training NN

[6]Data augmentation is often viewed intuitively as an addition of carefully perturbed synthetic data to the existing data set. Formally, however, it is a regularization method that encourages fitting to relevant features of the data rather than to irrelevant noise.

models. First and foremost, their aim is to speed up training and reduce computational costs.[7] However, many of the below techniques involve replacing a true algorithm or object with some reasonable approximation, and these techniques' effectiveness is possible only if they (or the respective downstream algorithms) are invariant to perturbations and robust to random noise that necessarily show up. In other words, along the way these speed-up techniques also act as regularizers.

The loss minimization algorithms in the existing literature are usually variations of a standard gradient descent with some stochasticity added (for an early source, see Robbins and Monro, 1951; for a Bayesian interpetation see Mandt et al., 2017).

The granularity of data inputs used for gradient computations and parameter updates vary from utilizing the full data set ("batch gradient descent") to several ("mini-batch gradient descent") or single ("stochastic gradient descent") randomly chosen block(s) of observations (that is, training example(s)). Smaller batches prevent using the most efficient computational tools and exhibit noisier convergence; but such built-in stochasticity helps avoiding local minima, hence works as a regularizer (see Wilson and Martinez, 2003).

Data standardization is useful in many statistical learning problems, but in the context of NNs it is even more critical. One development of the general standardization idea that is tailored for NNs is separately standardizing the inputs to intermediate hidden layers ("batch normalization", see Ioffe and Szegedy, 2015).

Minimization algorithms make use of efficient ways for computing gradients that improve upon the canonical chain rule ("backward propagation of errors", or just "backpropagation", see Rumelhart et al., 1986).

There are also methods used for ensuring minimization algorithms' satisfactory convergence, i.e. one that is fast and avoids bad local minima. These include clever ways of updating parameters in the face of stochasticity inherent to mimimization algorithms used: the learning rate is high far from the minimum in order to ensure fast learning, but falls as the minimum is approached to prevent overshooting ("learning rate decay", see Robbins and Monro, 1951; Welling and Teh, 2011); paramater updating maintains memory of recent updates in order to keep improving in the promising direction ("momentum", see Bengio et al., 2012; Sutskever et al., 2013). Also, simple as it is, an outright termination of the algorithm's iterations well before the minimum is reached saves on computational costs with little downside ("early stopping", see Prechelt, 1997). Importantly, all of the above methods contribute to regularization as well.

Many of the methods above have their own hypermarameters (random seeds, dropout probability, learning rates, etc.). The hyperparameters' initial values—and sometimes modification schedules too—are also important choices to be made.

[7]Hence, such techniques are often just practical tricks that are employed only out of necessity and computational resource constaints.

"Overparameterization": A common question about NNs (e.g., see Zhang et al., 2017; Belkin et al., 2018) is concerned with the fact that in overwhelming majority of applications the number of NN parameters exceeds the number of training observations by several orders of magnitude — what are the underlying mechanisms that allow NNs with large explanatory capacity that fit arbitrarily accurately on the training sample also to perform well in the test sample, i.e. to properly generalize?

Formally, the basic explanation is that all these parameters are not entirely free. For instance, there are some trivial symmetries and equivalences in the space of NN weights (Chen et al., 1993; Kurková and Kainen, 1994). Moreover, parameters of properly trained NNs turn out to lie in a lower-dimensional subspace (due to "noise stability" property of Arora et al., 2018; also see Hochreiter and Schmidhuber, 1997, Hinton and Van Camp, 1993, for a related argument).

3.2 Data

Our raw data set comprises: Real GDP, seasonally adjusted annual rate, and GDP Price Deflator, seasonally adjusted (both at quarterly frequency, sourced from Federal Reserve)[8]; Equal Weight Continuous Commodity Index (at daily frequency, sourced from Thomson Reuters); Effective Fed Funds Rate and Total Nonborrowed Reserves (at monthly frequency, sourced from Federal Reserve). Daily frequency data are aggregated into monthly by averaging, while quarterly frequency data are disaggregated into monthly using a linear state-space model (as in Bernanke et al., 1997). Next, data are transformed as follows: Real GDP, GDP Deflator and Commodity Price Index into month-over-month growth rates; Nonborrowed Reserves into monthly differences. (Additionally, to prevent numerical complications, the training itself is done in terms of data series that are standardized to have a mean of 0 and a variance of 1.)

See Figures 3 and 4 in Appendix §A for visual representations of monthly-frequency data prior and after the transformations, respectively.[9]

The resulting data set covers the time period from 1959:M2 to 2018:M03.

3.3 Implementation

We split those 59 years of monthly data into 3 sub-samples:

(i) training sample used for parameter learning (1959:M2 to 2013:M3);

(ii) validation sample used for adjusting the learning rate in oder to prevent data memorization (2013:M4 to 2015:M9);

[8]Seasonally adjusted data are used for simplicity, because ANNs' treatment of seasonality in a small-sample regime as here is an important research problem in its own right and deserves a separate study.

[9]Notice the effect of quantitative easing policy on Nonborrowed Reserves at the end of the sample period.

(iii) testing sample that the NN does not see until we are finished with learning and start benchmarking NN's usefulness (2015:M10 to 2018:M03).

Here we have sequential data, i.e. each observation is dependent on other observations immediately prior to it. Hence, one training example includes several (specifically, 24) consequtive observations: $\{\boldsymbol{y}_{t-\ell}\}_{\ell=0}^{23}$.

Because of data dependence, predictions over the validation and testing sub-samples are made multiple steps ahead (splitting each sub-sample in half and predicting 15 steps ahead, thus reserving 2×15 months for validation and 2×15 months for testing).

We use the "golden-standard" mini-batch gradient descent, with 4 training examples per batch. Then, 1 training epoch is composed of a full run through the training data set, thus feeding the training examples for all possible t: $\{\{\boldsymbol{y}_{t-\ell}\}_{\ell=0}^{23}\}_{t=24}^{650}$.

Our NN has a general LSTM architecture introduced earlier. More specifically, NN cells in the visible input layer take in the 5-dimensional data vector (so, $K := 5$) and pass it on untransformed. The cells in the hidden layers use Rectified Linear Units (ReLU) as the activation functions applied to their inputs. The cells in the visible output layer apply simple linear transformations as activation functions on their inputs from the last hidden layer; with the Mean Squared Error loss function ultimately used in our NN, in this final visible layer we are effectively just running Ordinary Least Squares regressions on the layer's inputs.

Within certain range, we try architectures of varying depth and width. We consider uniform architectures, as well as the ones with squeezed ("bottleneck" layer used for dimension reduction; see Rumelhart et al., 1985, and Bourlard and Kemp, 1988; also see usage examples in Hinton and Salakhutdinov, 2006, and Szegedy et al., 2014) and expanding shapes ("sparse" representation; see Ranzato et al., 2007; also see usage example in Delahunt and Kutz, 2018).

The minimization algorithm used is adaptive moment estimation, or Adam (Kingma and Ba, 2014). Each architecture is trained for 20 epochs, with a decaying learning rate. Lastly, the dropout probability is set at 0.2.[10]

To sum up, our implementation recipe has four stages: (i) set hyperparameters (which requires some tuning); (ii) define specification, train and regularize it; (iii) repeat the previous step for a range of specifications; (iv) choose specification with best training fit.[11]

[10] Additional hyperparameter settings are available upon request.

[11] We do not use as a criterion for choosing the best NN specification the validation sample fit (because our validation set is too small for that) or the best training epoch (because learning rate decay makes this extraneous).

3.4 Benchmark

For comparison, we use a standard Vector Auto-Regressive model. VAR model is defined by the following vector-valued equation:

$$\hat{\boldsymbol{y}}_t = \boldsymbol{a}_0 + \boldsymbol{A}_1 \boldsymbol{y}_{t-1} + \ldots + \boldsymbol{A}_L \boldsymbol{y}_{t-L}, \tag{6}$$

where $\boldsymbol{y}_{t-\ell}$ are lags of a K-dimensional random variable $\boldsymbol{y}_t$; while $\boldsymbol{a}_0 \in \mathbb{R}^K$ and $\boldsymbol{A}_\ell \in \mathbb{R}^{K \times K}$ are parameters; with lag $\ell \in \{1, \ldots, L\}$; and $\hat{\boldsymbol{y}}_t$ on the left-hand side has the same interpretation as before in equation (2).

More compactly, for $\boldsymbol{B} := [\boldsymbol{\alpha}, \boldsymbol{A}_1, \ldots, \boldsymbol{A}_L]$ and $\boldsymbol{z}_t := [1; \boldsymbol{y}_{t-1}; \ldots; \boldsymbol{y}_{t-L}]$, and following the notation from (4), this becomes

$$\hat{\boldsymbol{y}}_t = \boldsymbol{B}\boldsymbol{z}_t = \mathrm{id}^B(\boldsymbol{z}_t), \tag{7}$$

with the identity function defined as

$$\mathrm{id}(\boldsymbol{z}) := \boldsymbol{z}.$$

From this perspective, it becomes clear that VAR can be seen as a restricted version of a more general multivariate NN model (with MRNN and LSTM MRNN adding further enrichments).

We apply this model to the data set described earlier. Thus, $K = 5$ again. We split the sample the same way as before into training (1959:M2 to 2013:M3), validation (2013:M4 to 2015:M9) and testing (2015:M10 to 2018:M03) sub-samples. Parameters are estimated by Ordinary Least Squares.

Importantly, we follow the "predictive" approach from above and choose the optimal lag length L by comparing the predictive performance on the validation set. Such an explicitly predictive approach differs from the more traditional lag selection practice in VAR literature that is based on various information criteria: the latter also hopes to achieve the generalization ability, albeit indirectly and implicitly. (In the interest of transparency, below we also provide the results for the traditional approach: it selects the lag length using Akaike Information Criterion, combining the training and validation sub-samples together into a larger estimation sample.)

4 Results

4.1 All nets

After tuning the hyperparametes, we trained and regularized a range of architectures, from a modest NN with just 1 hidden layer of 5 units to a fairly rich one comprising 4 hidden layers with 500 units in each. Table 1 summarizes their fitting and prediction performance, measured in terms of Mean Squared Errors for standardized series, over the training, validation and testing samples.

Table 1: Long Short-Term Memory Multivariate Recurrent Neural Nets, Prediction Results

I				II				III				IV			
n_1	Train.	Valid.	Test.	n_1-n_2	Train.	Valid.	Test.	n_1-n_2-n_3	Train.	Valid.	Test.	n_1-n_2-n_3-n_4	Train.	Valid.	Test.
5	0.7448	5.0545	4.1719	5-5	0.7151	5.6903	4.1039	5-5-5	0.7544	5.1885	4.0098	5-5-5-5	1.1079	5.8338	4.4154
								5-10-5	0.7827	5.2218	4.0399	5-10-10-5	0.7800	5.3570	4.1138
								10-5-10	0.7173	5.1455	3.8792	10-5-5-10	0.8601	5.8477	4.5416
10	0.6880	5.2903	3.9699	10-10	0.6311	5.1148	4.0858	10-10-10	0.6848	5.1855	4.0427	10-10-10-10	0.8211	5.5939	4.2782
								10-25-10	0.6865	5.3224	5.8654	10-25-25-10	0.7850	5.4949	3.9825
								25-10-25	0.6805	4.9532	4.0537	25-10-10-25	0.8244	5.0129	3.9364
25	0.6818	5.1748	3.9815	25-25	0.6431	5.6948	3.9727	25-25-25	0.6885	5.0445	4.3594	25-25-25-25	0.7851	5.1068	4.0044
								25-50-25	0.6596	5.0147	4.2529	25-50-50-25	0.6842	5.0778	4.2164
								50-25-50	0.6624	5.4168	4.9084	50-25-25-50	0.7395	4.8835	4.4527
50	0.6530	5.1150	3.8830	50-50	0.7071	5.2178	4.4503	50-50-50	0.6679	5.1518	4.0167	50-50-50-50	0.9126	5.9132	4.6092
								50-100-50	0.6462	5.1015	4.1877	50-100-100-50	0.6510	5.3037	4.0904
								100-50-100	0.7986	5.8369	4.5926	100-50-50-100	0.8387	5.2875	4.0926
100	0.6913	5.0546	4.7636	100-100	0.7245	5.1625	4.2298	100-100-100	0.6889	5.5366	4.7786	100-100-100-100	0.6723	5.1201	4.1073
								100-200-100	0.8184	5.3113	4.3729	100-200-200-100	0.7485	8.4162	4.4783
								200-100-200	0.7569	5.3580	4.4943	200-100-100-200	0.8014	5.3540	4.1195
200	0.7320	5.0373	4.1997	200-200	0.7631	4.9071	4.6773	200-200-200	0.7294	6.3925	7.0641	200-200-200-200	0.6501	5.3566	4.6297
								200-300-200	0.7934	4.9213	4.2428	200-300-300-200	0.6947	5.2851	4.1295
								300-200-300	0.8057	5.7319	5.4270	300-200-200-300	1.1478	7.9611	6.8660
300	0.7812	4.9054	3.8704	300-300	0.7920	4.9043	4.0495	300-300-300	0.7999	6.0067	4.3594	300-300-300-300	0.7528	7.6135	9.8760
								300-400-300	0.8138	5.2597	4.4740	300-400-400-300	0.6722	5.6456	5.0593
								400-300-400	0.8197	5.1196	4.8146	400-300-300-400	0.7133	15.5906	10.3539
400	0.7879	5.1756	3.9187	400-400	0.7916	4.9197	4.1388	400-400-400	0.9105	5.4103	7.7445	400-400-400-400	0.6499	4.7265	4.1539
								400-500-400	0.7620	5.1613	4.7135	400-500-500-400	0.7153	7.8092	8.9807
								500-400-500	0.8272	4.8364	5.0482	500-400-400-500	0.6600	5.0797	4.5522
500	0.8375	5.0924	3.9175	500-500	0.8012	4.7628	4.1674	500-500-500	0.8078	5.2696	4.7725	500-500-500-500	0.8630	12.6689	5.8035

Notes: Panels I to IV present the results for NNs of different depth, from 1 to 4 hidden layers, respectively. The rows present the results for NNs with different numbers and permutations of units, from $n_1 = 5$ in one hidden layer to $[n_1, n_2, n_3, n_4] = [500, 500, 500, 500]$ in, correspondingly, hidden layers 1, 2, 3, 4. For each architecture, training ran for 20 epochs. MSEs (calculated for standardized series, averaged over variables and time periods) in the training, validation and testing sub-samples are given; predictions are for multiple steps ahead in the latter two sub-samples. Additional technical details about the implementation are provided in the main text.

Evidently, a lot of architectures achieve a fairly similar performance. (Also, training results are sensitive to the random seed used, due to the nature of the problem as well as that of the optimization techniques.) Such a situation usually calls for ensemble averaging approaches. However, given that we are already using dropout regularization, which effectively conducts a kind of model averaging, we will abstain from additional mixing.

Moreover, notice that specifications with larger capacities not only exhibit good training sample fit, but also do not necessarily underpeform in their testing sample predictions. This corroborates the effectiveness of our regularization techniques.

4.2 Best net

4.2.1 Forecasting

We take the trained NN instance with the lowest training sample MSE as our preferred NN architecture.

With the MSE of 0.6311, this turns out to be a fairly minimalistic specification featuring 2 layers, each including just 10 units. Under the hood, such a specification comprises 1535 NN parameters in total (refer to equations 1 and 5):

- (10+50+100) + (10+50+100) + (10+50+100) + (10+50+100) = 640 parameters in the first hidden layer;
- (10 + 100 + 100) + (10 + 100 + 100) + (10 + 100 + 100) + (10 + 100 + 100) = 840 parameters in the second hidden layer; and
- (5 + 50) = 55 parameters in the visible output layer.

The validation sample predictive fit is 5.1148 (multiple-steps-ahead, accuracy measured in MSE terms). This is worse than in some of the alternative trained architectures; the minimum is 4.7265 for a 4-layered NN with 400 units in each. However, given that our validation sample is very small due to data limitations, it is impossible to tell apart an underperformance of the specific NN instance and a peculiarity of the chosen validation data sample.[12]

The testing sample was not available for our preferred NN at the training stage: its predictive fit ended up being 4.0858 there. Many alternative NNs performed even better, with the minimum MSE being 3.8704 for a 1-layered NN with 300 units; but the testing sample and the corresponding predictive performances became known only ex-post.

We are going to compare our best NN with the benchmark model, Vector Auto-Regression. Predictive performance on the validation sample suggested using 4 lags, i.e. VAR(4). Such specification comprises 105 parameters in total (refer to equation 6): (5+25+25+25+25). As can be seen from Table 2, VAR model's in-sample performance dominates that of our preferred NN: the training sample MSE is 0.5128, and the validation

[12]That is why we relied on the—wider—training sample's fit for architecture choice.

Table 2: Vector Auto-Regression, Prediction Results

L	Training	Validation	Training&Validation	Testing
4	0.5128	4.2544		4.9624
11			0.5372	5.7299

Notes: The results are presented for VARs with lag lengths $L = 4$ (selected basing on validation sample performance) and $L = 11$ (selected basing on AIC). Estimation was done by OLS. MSEs (calculated for standardized series, averaged over variables and time periods) in the training, validation and testing as well as combined testing & validation sub-samples are given; predictions are for multiple steps ahead in the validation and testing sub-samples. Additional technical details about the implementation are provided in the main text.

sample MSE is 4.2544. However, VAR underperforms out-of-sample: for the test data, its MSE of 4.9624 is about 20% worse. (For comparison, Akaike Information Criterion suggested using 11 lags, the resulting VAR(11) model in Table 2 performs poorly out-of-sample, its test-data MSE is 5.7299; we prefer the VAR(4) specification as a benchmark.)

Let us zoom closer into two models' predictions on Figure 1. Neither of the two captures high-frequency movements, instead they are trying to pin down the movements in conditional means. Failure to achieve the former, much more challenging task should not be surprising: the time series in question are inherently noisy (just think of unpredictable weather conditions), they are affected by many other factors beyond the 5-variable economic system that is being modeled (think of housing market dynamics or global economic environment).

One point to keep in mind is that for most practical purposes, what matters is the cumulative forecast over several periods. In such a situation, much of the high-frequency noise cancels out. Moreover, even a small improvement in prediction accuracy can be very valuable: say, an improvement of merely 0.1 percentage point per month produces more than 1 percentage point improvement in the case of GDP growth forecast on a 12-months horizon.

Another point concerns our standardization convention. We conduct the modeling in terms of standardized time series, each having the variance of 1, and weight them equally in the mean-squared loss function. So, in the original scaling some of the forecasts on Figure 1 may look farther off the target than they are in the standard-deviations space that is important from the modeling perspective. E.g., NN's Fed Funds rate predictions are not as bad as they may seem in the original scaling. For the reference, the standard deviations of our 5 variables [Real GDP growth, GDP Deflator inflation, Commodity Price inflation,

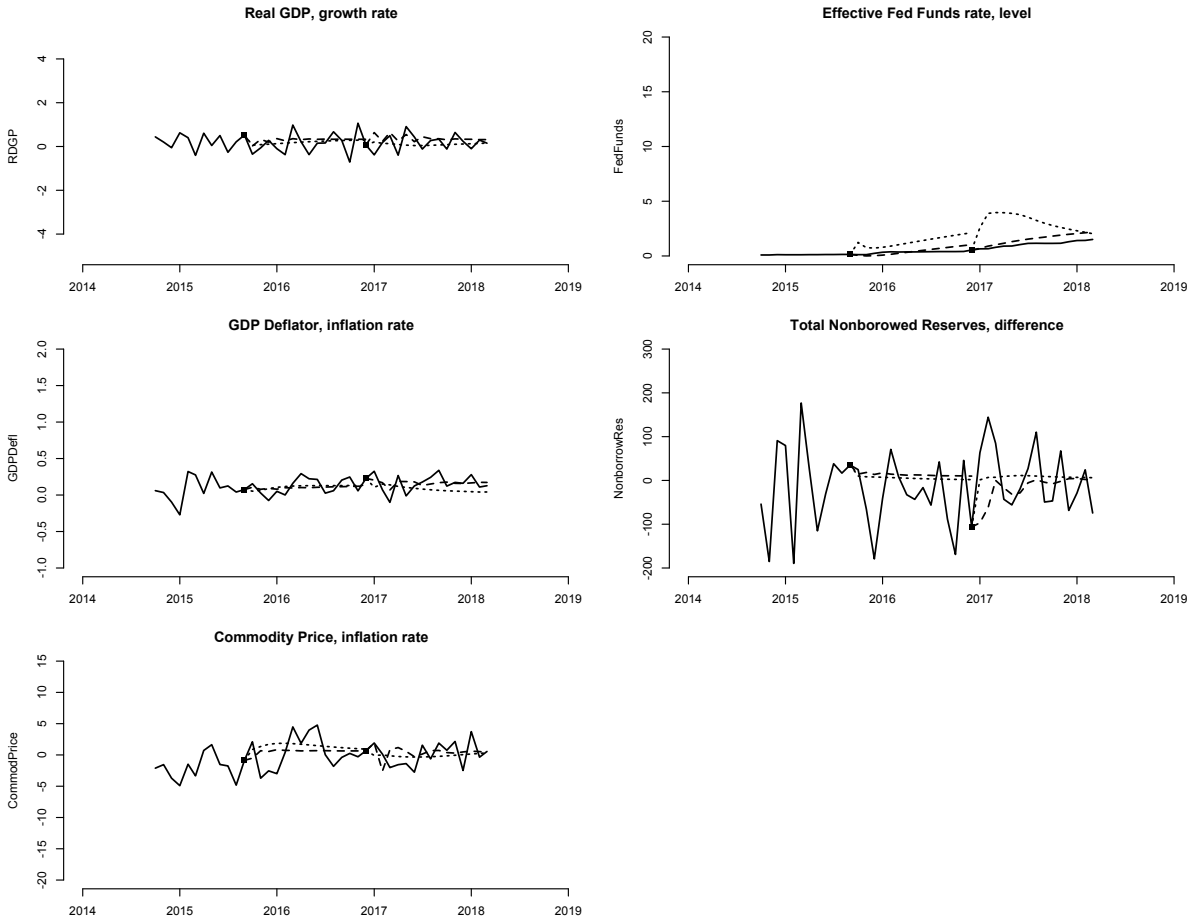

Figure 1: Predictive performance.
Multiple-steps-ahead predictions of the MRNN (dotted) and VAR (dashed) models over the testing sample: first, period 2015:M10 to 2016:M12, and second, period 2017:M1 to 2018:M3; with the starting points, 2015:M9 and 2016:M12, marked (by solid squares); together with the historical data (solid) from 2014:M10 to 2018:M3.

Fed Funds rate, Nonborrowed Reserves change] are [0.91, 0.22, 2.76, 3.53, 26.16]; thus the magnitudes of the plotted prediction errors in Fed Funds rate and Nonborrowed Reserves change are overblown, and those of GDP Deflator inflation are underemphasized.[13]

Overall, the take-away is that, at least in our exercise, MRNN outperformed VAR in out-of-sample prediction (and, thus, in generalization).

4.2.2 Impulse responses

Beyond forecasting, time-series models can be used for measuring system reactions to shocks. For example, if commodity prices (think of oil) jump sharply up, what would happen to other economic variables?

Technically, impulse responses of a multivariate time-series system are defined as movements (responses) of all variables once the equilibrated system receives a single-period shock (impulse) to one of the variables. (Before proceeding further, note that in this paper we only consider temporary shocks, and completely ignore permanent ones; we also ignore potential asymmetry and thus do not examine separately impulses in the opposite directions.)

In VAR, this is performed as follows. Given the covarying nature of the system variables $\boldsymbol{y}$, and thus of the shocks to these variables $\boldsymbol{\xi}$, a correct way to generate a single shock has to account for the covariance structure. Perhaps the simplest way of doing this is using a Cholesky decomposition of the shocks' variance-covariance matrix: $\boldsymbol{\Sigma} = \boldsymbol{L}\boldsymbol{L}^{\intercal}$, where $\boldsymbol{L}$ is the resulting lower-triangular matrix. (Technically, this allows to identify contamporaneous links between variables in our reduced-form VAR using recursive indentification restrictions.) As a result, it holds that $\boldsymbol{\xi} = L\boldsymbol{\varepsilon}$, given some fundamental uncorrelated shock $\boldsymbol{\varepsilon}$ that has an identity variance-covariance matrix $\boldsymbol{I}$. Thus, at impulse time τ a shock to variable k is cleanly injected by (i) perturbing the k-th element of $\boldsymbol{\varepsilon}$: $\boldsymbol{\varepsilon}_\tau := \boldsymbol{e}_k$, where $\boldsymbol{e}_k$ is a column vector with 1 in row k and 0 in all other rows; and then (ii) adding the manufactured shock $\boldsymbol{\xi}_\tau := \boldsymbol{L}\boldsymbol{\varepsilon}_\tau$ to the (previously equilibrated) system: $\boldsymbol{y}_\tau := \bar{\boldsymbol{y}} + \boldsymbol{\xi}_\tau$. Instantaneously in the same period τ, the shock percolates to the variables with non-zero elements of lower triangular matrix $\boldsymbol{L}$; from period $\tau+1$ onward, the whole system is affected according to $\boldsymbol{y}_\tau$'s law of motion.

However, in LSTM MRNN a similarly straightforward approach as above is not applicable. Given that LSTM networks comprise not only observed variables $\boldsymbol{y}$, but also unobserved state variables $\boldsymbol{c}$ and $\boldsymbol{h}$, the shocks should perturb both the observed as well as the unobserved variables.[14] Ideally, this should be done by a careful injection of shocks

[13] Interestingly, as can be seen in Figure 1, on the basis of past history MRNN predicts a more agressive tightening of the Fed's policy rate, and, correspondingly, lower inflation rates than we actually saw happening lately.

[14] Basically, what should be shifted is the shocked variable's conditional mean, which is (loosely speaking) determined by the corresponding unobserved state variables. Otherwise, in our practice the shocked variable and the whole system just reverted to the unconditional mean almost immediately.

to the state variables (and/or having imposed restrictions on the relevant connection parameters). But this is a challenging task given the complexity of a typical ANN's network of hidden units and connections. For instance, generally there is no single state variable devoted to the variable that is to be shocked, the relevant elements can be scattered anywhere between and within the hidden units.

One indirect and naïve way that seems to work in practice is as follows. The shock $\boldsymbol{\xi}_\tau$ is manufactured using Cholesky decomposition as above. During the usual "warmup session" that is necessary for initialization of the trained LSTM before any usage (and in our case is carried out by feeding the NN with variables' training-sample means), in the last 12 periods of the session the shock $\boldsymbol{\xi}$ is added on top of the usual "warmup" inputs, thus shifting the corresponding state variables in the appropriate direction. Then, at impulse time τ, the shock is added, $\boldsymbol{y}_\tau := \bar{\boldsymbol{y}} + \boldsymbol{\xi}_\tau$, and the system evolves as above.

Reactions of the system to a temporary 1-standard-deviation increase in the Fed Funds rate (i.e., $\boldsymbol{\varepsilon}_\tau := \boldsymbol{e}_4$) are presented on Figure 2. The following observations can be made:[15]

(i) Real output growth accelarates at first, but then slows down in VAR model; it starts slowing down immediately in MRNN model. Output contraction/slowdown agrees with economic intuition (monetary contraction reduces nominal aggregate demand, affecting not only nominal but also real quantities).

(ii) Output inflation rate shoots up and only gradually returns to its long-run mean producing inflation acceleration in VAR; it rises in the short run (for several months), but then falls below the long-run mean producing inflation slowdown in MRNN. Higher inflation in response to Fed Funds increase is counter-intutive according to conventional economic theory (monetary contraction reduces nominal aggregate demand, initiating deflationary pressure); but agrees with (Neo-)Fisherian views, at least in the long-run for a permanent Fed Funds rate increase (since real interest rate is determined by economic fundamentals and can not be easily manipulated, higher nominal interest rate ultimately translates into higher inflation rate).

(iii) Commodities' inflation rate wobbles at first, and then stays at the rate below long-run mean for many years in VAR; it slows down right away and stays low in MRNN. Lower commodity prices when monetary policy is contractionary agrees with economic theory (tighter monetary supply increases the relative price of currency in terms of goods).

(iv) Fed Funds rate—after the initial shock—gradually reverts to its long-run mean in VAR; it reverts to the long-run mean in MRNN as well, albeit with some undershooting along the way. Gradual normalization following the initial impulse is a feature of any stable dynamic system.

[15]Henceforth, we use the wording "short run" and "long run" in terms of presented impulse response horizons, rather than in the sense of, correspondingly, temporary and permanent effects.

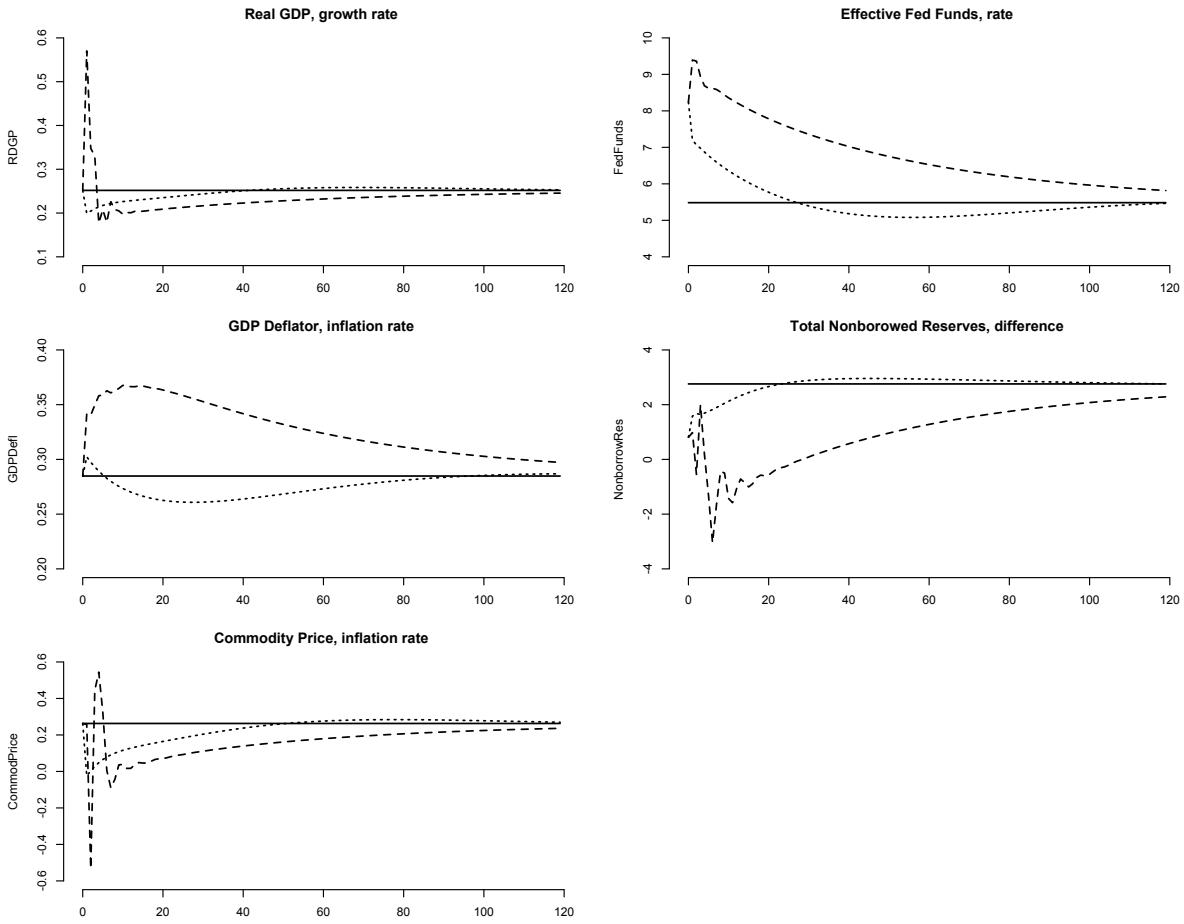

Figure 2: Impulse response functions.
Reactions in the MRNN (dotted) and VAR (dashed) models to a shock of +1 standard deviation in the Effective Fed Funds rate variable, together with the training/estimation sample means (solid). Units on the horizontal axis: months.

(v) Nonborrowed reserves contract right away and then gradually return to the long-run mean in VAR; they contract in MRNN as well. Reduction in the reserves banks hold with the central bank following a monetary contraction is in line with economic theory.

Apart from relatively mild quantative differences, the only stark qualitative difference between MRNN and VAR demonstrated in this exercise concerns the response of GDP Price Deflator.

Conventional economic theory predicts that Fed Funds rate increase leads to lower inflation (see e.g. Christiano et al., 2005), even though not necessarily materializing in the short run. The so-called "price puzzle", implying positive response of inflation to monetary policy rates, has been first noted by Sims (1992). Apart from the effect of some omitted variables, the "mechanistic" interpretation of this finding is that simple, theoretically unrestricted VARs "confuse" correlation with causation: historically, a high

rate of inflation has been associated with high nominal interest rates, and while higher inflation indeed causes a central bank to increase the policy interest rate (one direction of causality), such an increase itself causes inflation rate to subside (another, opposite direction of causality).[16] With the aim of clarifying contemporaneous causal links between variables in VARs, a large literature on structural identification restrictions has been born (leading to so-called structural VARs): it includes approaches based on formal theoretical models (see e.g. King et al., 1991; Galí, 1992; Bernanke and Mihov, 1998) as well as those based on more agnostic theoretical assumptions (for example, sign restrictions, as in Uhlig, 2005; Antolín-Díaz and Rubio-Ramírez, 2018; Uribe; 2018).

On the other hand, (Neo-)Fisherian thinking (Cochrane, 2018; Williamson, 2016) predicts that Fed Funds increase leads to higher inflation, at least in the long run in the case of a permanent Fed Funds rate increase. Strictly speaking, though, the above VAR result was obtained for a temporary Fed Funds rate rise.

Finally, with a caveat regarding temporary nature of monetary shock here, our result from MRNN implies the following: a small Neo-Fisherian effect in the short run; and a larger conventional effect in the long run, with the "correct" direction of causality identified this time without any structural restrictions.

4.2.3 Data augmentation

As a robustness check, in particular to address possible concerns about relying on a very limited data set, we attempt a data augmentation exercise.

Since in our modeling approach we abstain from imposing any structure on our data, the augmentation strategy we chose is noise contamination. Specifically, we (i) divide the existing (standardized) time series into blocks of 50 consequtive observations; (ii) copy the data set 10 times; (iii) randomly shuffle the blocks in the enlarged data set; (iv) add Gaussian noise (with zero mean and a variance-covariance observed in the training sample) scaled by 0.1; (v) additionally paste the last block of the (uncontaminated) training data set at the end of the time series. In the end, we extend the training data set from 650 to 6550 observations, leaving the validation and test samples untouched.

The MRNN on augmented data set was trained exactly the same way as before (and also for 20 epochs). When we applied the trained model to the uncontaminated data set, the measures of fit on all three sub-samples underperfomed those of our original MRNN: 0.6465 vs. 0.6311 in the training sample, 5.1902 vs. 5.1148 in the validation sample, and 4.4893 vs. 4.0858 in the testing sample.

[16] Identifying causal relations and the notion of causality itself is a profound question, we do not even attempt here to give it due justice. Two perspectives most relevant to social sciences are presented in Rubin (2005) or Morgan and Winship (2007) (potential outcomes framework) as well as in Pearl (2000) (causal calculus formulated in terms of Directed Acyclic Graphs). In natural sciences, experimental findings such as—taking just a few—Goswami et al. (2018) or Ringbauer et al. (2016) pose further theoretical challenges (for example, see Brukner, 2014; Allen et al., 2017).

Our data augmentation approach did not seem to offer any advantage over using just the existing data as is. However, a different approach may prove to be more fruitful, hence verification of different augmentation settings or perhaps even trying a more structural augmentation strategy is still a worthwhile exercise.

5 Conclusion

In modeling half a century of US economic time-series data, Multivariate Recurrent Neural Networks outperformed mainstream Vector Auro-Regressions in terms of forecasting accuracy and interpretability of their results. Higher precision of its out-of-sample predictions suggests that MRNN acquires stronger generalization ability. Moreover, at least in this particular exercise, MRNN appears to learn the "correct" (i.e., one that agrees with conventional economic theory) causal effect from the raw data, without us imposing any structural identification restrictions. Arguably, non-linear architecture of NNs and the prominence of predictive criterion in their training, as well as Long Short-Term Memory MRNN's capability to account for conditional dynamics of the modeled system's state variables make them well equipped for such tasks.

A Data inputs

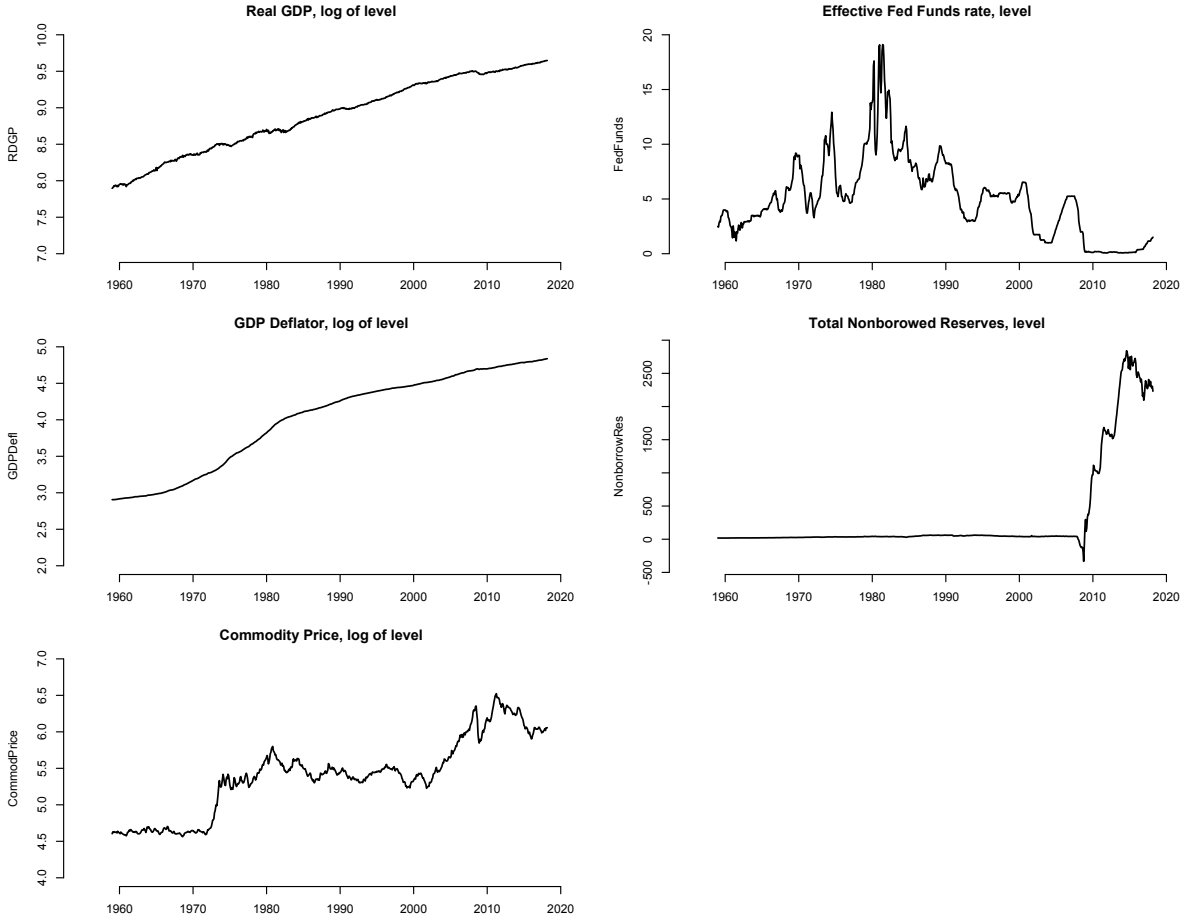

Figure 3: Plots of the initial data inputs.
Variables included: logarithm of Real GDP, logarithm of GDP Price Deflator, logarithm of Commodity Price Index, Effective Fed Funds Rate and Total Nonborrowed Reserves.
Time period covered: 1959:M1 to 2018:M03.

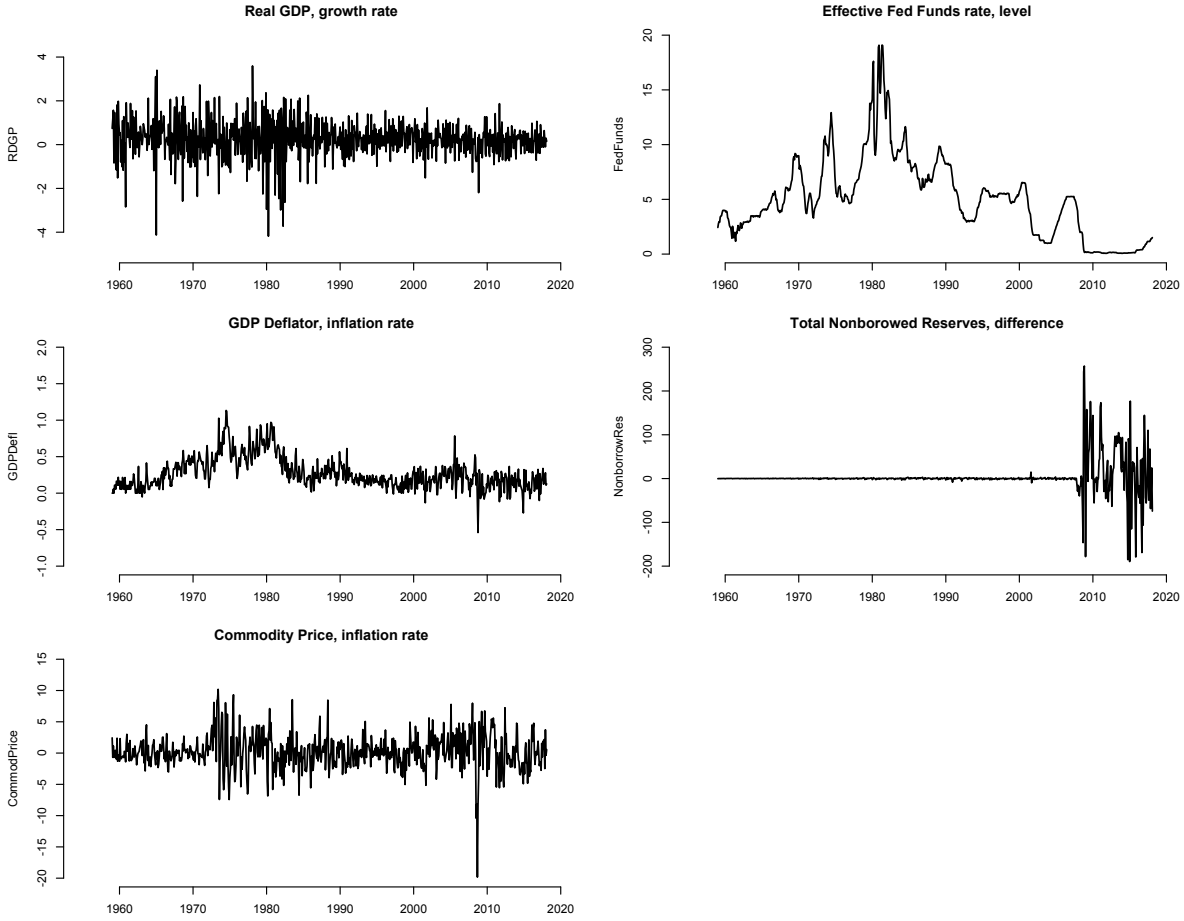

Figure 4: Plots of the transformed data inputs.
Variables included: Real GDP MoM growth rate, GDP Price Deflator MoM growth rate, Commodity Price Index MoM growth rate, Effective Fed Funds Rate and Total Nonborrowed Reserves monthly difference.
Time period covered: 1959:M2 to 2018:M03.

References

[1] Akaike, Hirotugu. (1974) "A New Look at the Statistical Model Identification", *IEEE Transactions on Automatic Control*, 19(6): 716–723.

[2] Allen, John-Mark A., Jonathan Barrett, Dominic C. Horsman, Ciarán M. Lee, and Robert W. Spekkens. (2017) "Quantum Common Causes and Quantum Causal Models", *Physical Review X*, 7: 031021.

[3] Antolín-Díaz, Juan, and Juan F. Rubio-Ramírez. (2018) "Narrative Sign Restrictions for SVARs", *American Economic Review*, 108(10): 2802–2829.

[4] Arnold, V. I. (1957) "On Functions of Three Variables", *Proceedings of the USSR Academy of Sciences*, 114: 679–681.

[5] Arora, Sanjeev, Rong Ge, Behnam Neyshabur, Yi Zhang. (2018) "Stronger Generalization Bounds for Deep Nets via a Compression Approach", arXiv:1802.05296.

[6] Belkin, Mikhail, Siyuan Ma, Soumik Mandal. (2018) "To Understand Deep Learning We Need to Understand Kernel Learning", arXiv:1802.01396.

[7] Bengio, Yoshua, Nicolas Boulanger-Lewandowski and Razvan Pascanu. (2012) "Advances in Optimizing Recurrent Networks", arXiv:1212.0901.

[8] Bernanke, Ben S., Mark Gertler, and Mark Watson. (1997) "Systematic Monetary Policy and the Effiects of Oil Price Shocks", *Brookings Papers on Economic Activity*, 1: 91–157.

[9] Bernanke, Ben S., and Ilian Mihov. (1998) "Measuring Monetary Policy", *Quarterly Journal of Economics*, 113(3): 869–902.

[10] Bishop, Christopher M. (2006) *Pattern Recognition and Machine Learning*, Springer.

[11] Bourlard, Hervé, and Yves Kamp. (1988) "Auto-association by Multilayer Perceptrons and Singular Value Decomposition", *Biological Cybernetics*, 59(4-5): 291–294.

[12] Box, George, Gwilym Jenkins. (1970) *Time Series Analysis: Forecasting and Control*, San Francisco: Holden-Day.

[13] Brukner, Časlav. (2014) "Quantum Causality", *Nature Physics*, 10: 259–263.

[14] Cagli, Eleonora, Cécile Dumas, and Emmanuel Prouff. (2017) "Convolutional Neural Networks with Data Augmentation Against Jitter-based Countermeasures", Proceedings of the 19th International Conference Cryptographic Hardware and Embedded Systems, 45–68.

[15] Chen, An Mei, Haw-minn Lu and Robert Hecht-Nielsen. (1993) "On the Geometry of Feedforward Neural Network Error Surfaces", *Neural Computation*, 5(6): 910–927.

[16] Christiano, Lawrence J., Martin Eichenbaum, and Charles L. Evans. (2005) "Nominal Rigidities and the Dynamic Effects of a Shock to Monetary Policy", *Journal of Political Economy*, 113(1): 1–45.

[17] Cochrane, John H. (2018) "Michelson-Morley, Fisher, and Occam: The Radical Implications of Stable Quiet Inflation at the Zero Bound", *NBER Macroeconomics Annual*, 32(1): 113–226.

[18] Cui, Xiaodong, Vaibhava Goel and Brian Kingsbury. (2015) "Data Augmentation for Deep Neural Network Acoustic Modeling", *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(9): 1469–1477.

[19] Cybenko, George. (1989) "Approximation by Superpositions of a Sigmoidal Function", *Mathematics of Control, Signals, and Systems*, 2: 303–314.

[20] Delahunt, Charles B., and J. Nathan Kutz. (2018) "Insect Cyborgs: Biological Feature Generators Improve Machine Learning Accuracy on Limited Data", arXiv:1808.08124.

[21] Dixon, Matthew F., Nicholas G. Polson, Vadim O. Sokolov. (2018) "Deep Learning for Spatio-Temporal Modeling: Dynamic Traffic Flows and High Frequency Trading", arXiv:1705.09851.

[22] Dütting, Paul, Zhe Feng, Harikrishna Narasimhan, David C. Parkes. (2017) "Optimal Auctions through Deep Learning", arXiv:1706.03459.

[23] Elman, Jeffrey L. (1990) "Finding Structure in Time", Cognitive Science 14(2): 179-211.

[24] Feng, Guanhao, Jingyu He, Nicholas G. Polson. (2018) "Deep Learning for Predicting Asset Returns", arXiv:1804.09314.

[25] Friston, Karl J., Richard Rosch, Thomas Parr, Cathy Price, Howard Bowman (2018) "Deep Temporal Models and Active Inference", *Neuroscience and Biobehavioral Reviews*, 90: 486–501.

[26] Galí, Jordi. (1992) "How Well Does the IS-LM Model Fit Postwar U.S. Data?", *Quarterly Journal of Economics*, 107(2): 709–738.

[27] Goel, Hardik, Igor Melnyk, Nikunj Oza, Bryan Matthews, Arindam Banerjee. (2016) "Multivariate Aviation Time Series Modeling: VARs vs. LSTMs", Manuscript.

[28] Goldberg, Yoav. (2016) "A Primer on Neural Network Models for Natural Language Processing", *Journal of Arti*

cial Intelligence Research, 57: 345–420.

[29] Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. (2016) *Deep Learning*, MIT Press.

[30] Goswami, K. , C. Giarmatzi, M. Kewming, F. Costa, C. Branciard, J. Romero and A. G. White. (2018) "Indefinite Causal Order in a Quantum Switch", *Physical Review Letters*, 121: 090503.

[31] Graves, Alex, and Jürgen Schmidhuber. (2009) "Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks", *Advances in Neural Information Processing Systems*, 21.

[32] Gu, Shihao, Bryan Kelly and Dacheng Xiu. (2018) "Empirical Asset Pricing via Machine Learning", Manuscript.

[33] Hamilton, James D. (1994) *Time Series Analysis*, Princeton: Princeton University Press.

[34] Hartford, Jason, Greg Lewis, Kevin Leyton-Brown, Matt Taddy. (2016) "Counterfactual Prediction with Deep Instrumental Variables Networks", arXiv:1612.09596.

[35] He, Yang-Hui. (2017) "Machine-learning the String Landscape", *Physics Letters B*, 774, pp. 564–568.

[36] Heaton, J. B., N. G. Polson, J. H. Witte. (2016a) "Deep Learning in Finance", arXiv:1602.06561.

[37] Heaton, J. B., N. G. Polson, J. H. Witte. (2016b) "Deep Portfolio Theory", arXiv:1605.07230.

[38] Hinton, Geoffrey E., and Drew van Camp. (1993) "Keeping Neural Networks Simple by Minimizing the Description Length of the Weights", Proceedings of the sixth annual conference on Computational learning theory: 5–13.

[39] Hinton, G. E., and R. R. Salakhutdinov. (2006) "Reducing the Dimensionality of Data with Neural Networks", *Science*, 313: 504–507.

[40] Hochreiter, Sepp, and Jürgen Schmidhuber. (1997) "Flat Minima", *Neural Computation*, 9(1): 1–42.

[41] Hochreiter, Sepp, and Jürgen Schmidhuber. (1997) "Long Short-Term Memory", *Neural Computation* 9(8): 1735–1780.

[42] Hornik, Kurt, Maxwell Stinchcombe and Halber White. (1989) "Multilayer Feedforward Networks are Universal Approximators", *Neural Networks*, 2: 359–366.

[43] Jaeger, Herbert, and Harald Haas. (2004) "Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication", *Science*, 304: 708–80.

[44] Jaitly, Navdeep, and Geoffrey E. Hinton. (2013) "Vocal Tract Length Perturbation (VTLP) Improves Speech Recognition", Proceedings of the 30 th International Conference on Machine Learning.

[45] Ioffe, Sergey, Christian Szegedy. (2015) "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift", arXiv:1502.03167.

[46] Ivakhnenko, A. G., and V. G. Lapa. (1965) *Cybernetic Predicting Devices*, CCM Information Corporation.

[47] King, Robert G., Charles I. Plosser, James H. Stock and Mark W. Watson. (1991) "Stochastic Trends and Economic Fluctuations", *American Economic Review*, 81(4): 819–840.

[48] Kingma, Diederik P., and Jimmy Ba. (2014) "Adam: A Method for Stochastic Optimization", arXiv:1412.6980.

[49] Ko, Tom, Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. (2015) "Audio Augmentation for Speech Recognition", INTERSPEECH: 3586–3589.

[50] Kolmogorov, A. N. (1957) "On the Representation of Continuous Functions of Many Variables by Superposition of Continuous Functions of One Variable and Addition", *Proceedings of the USSR Academy of Sciences*, 114, 953–956.

[51] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. (2012) "Imagenet Classification with Deep Convolutional Neural Networks", *Advances in Neural Information Processing Systems*, 25: 1097-1105.

[52] Kurková, Věra, and Paul C. Kainen. (1994) "Functionally Equivalent Feedforward Neural Networks", *Neural Computation*, 6(3):543–558.

[53] LeCun, Yann, Léon Bottou, Yoshua Bengio and Patrick Haffner. (1998) "Gradient-Based Learning Applied to Document Recognition", *Proceedings of the IEEE*, 86(11), 2278–2324.

[54] Lei, Na, Zhongxuan Luo, Shing-Tung Yau, David Xianfeng Gu. (2018) "Geometric Understanding of Deep Learning", arXiv:1805.10451.

[55] Liao, Qianli, and Tomaso Poggio. (2016) "Bridging the Gaps Between Residual Learning, Recurrent Neural Networks and Visual Cortex", arXiv:1604.03640.

[56] Lipton, Zachary C., John Berkowitz, Charles Elkan. (2015) "A Critical Review of Recurrent Neural Networks for Sequence Learning", arXiv:1506.00019.

[57] Maghrebi, Houssem, Thibault Portigliatti, Emmanuel Prouff. (2016) "Breaking Cryptographic Implementations Using Deep Learning Techniques", Proceedings of the 6th International Conference Security, Privacy, and Applied Cryptography Engineering, 3–26.

[58] Mandt, Stephan, Matthew D. Hoffman, David M. Blei. (2017) "Stochastic Gradient Descent as Approximate Bayesian Inference", arXiv:1704.04289.

[59] McCulloch Warren, S., and Walter Pitts. (1943) "A Logical Calculus of the Ideas Immanent in Nervous Activity", *Bulletin of Mathematical Biophysics*, 5: 115–133.

[60] Mehta, Pankaj, David J. Schwab. (2014) "An Exact Mapping Between the Variational Renormalization Group and Deep Learning", arXiv:1410.3831.

[61] Morgan, S., and Winship, C. (2007) *Counterfactuals and Causal Inference: Methods and Principles for Social Research*, New York: Cambridge University Press.

[62] Patel, Ankit B., Tan Nguyen, Richard G. Baraniuk. (2015)"A Probabilistic Theory of Deep Learning", arXiv:1504.00641.

[63] Pathak, Jaideep, Alexander Wikner, Rebeckah Fussell, Sarthak Chandra, Brian R. Hunt, Michelle Girvan, and Edward Ott. (2018) "Hybrid Forecasting of Chaotic Processes: Using Machine Learning in Conjunction with a Knowledge-based Model", *Chaos*, 28: 041101.

[64] Pearl, Judea. (2000) *Causality: Models, Reasoning, and Inference*, Cambridge University Press.

[65] Polson, Nicholas G., and Vadim O. Sokolov. (2017) "Deep Learning: A Bayesian Perspective", arXiv:1706.00473.

[66] Prechelt, Lutz. (1998) "Early Stopping — But When?" *Neural Networks: Tricks Of The Trade*, 1524: 55–69.

[67] Ranzato, Marc'Aurelio, Christopher Poultney, Sumit Chopra, and Yann LeCun. (2007) "Efficient Learning of Sparse Representations with an Energy-Based Model", *Advances in Neural Information Processing Systems*, 19.

[68] Ringbauer, Martin, Christina Giarmatzi, Rafael Chaves, Fabio Costa, Andrew G. White, Alessandro Fedrizzi. (2016) "Experimental Test of Nonlocal Causality", *Science Advances*, 2(8): e1600162.

[69] Robbins, H. and Monro, S. (1951) "A Stochastic Approximation Method", *Annals of Mathematical Statistics*, 22(3): 400–407.

[70] Rosenblatt, Frank. (1958) "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain", *Psychological Review*, 65(6): 386–408.

[71] Rubin, Donald B. (2005). "Causal Inference Using Potential Outcomes: Design, Modeling, Decisions", *Journal of the American Statistical Association*, 100(469): 322–331.

[72] Rumelhart, D. E., G. E. Hinton and R. J. Williams. (1985) "Learning Internal Representations by Error Propagation". In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1, pages 318–362.

[73] Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. (1986) "Learning Representations by Back-Propagating Errors", *Nature*, 323(6088): 533–536.

[74] Schmidhuber, Jürgen. (2015) "Deep Learning in Neural Networks: An Overview", *Neural Networks*, 61: 85–117.

[75] Schwartz-Ziv, Ravid, Naftali Tishby. (2017) "Opening the black box of Deep Neural Networks via Information", arXiv:1703.00810.

[76] Sietsma, Jocelyn, and Robert J. F. Dow. (1991) "Creating Artificial Neural Networks That Generalize", *Neural Networks*, 4: 67–79.

[77] Simard, Patrice Y., Dave Steinkraus, John C. Platt. (2003) "Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis", Proceedings of the Seventh International Conference on Document Analysis and Recognition.

[78] Sims, Christopher A. (1980) "Macroeconomics and Reality", *Econometrica*, 48(1): 1–48.

[79] Sims, Christopher A. (1992) "Interpreting the macroeconomic time series facts: The effects of monetary policy", *European Economic Review*, 36: 975–1000.

[80] Sirignano, Justin A. (2016) "Deep Learning for Limit Order Books", arXiv:1601.01987.

[81] Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov. (2014) "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", *Journal of Machine Learning Research*, 15: 1929–1958.

[82] Stock, James H, Mark W. Watson. (2016) "Dynamic Factor Models, Factor-Augmented Vector Autoregressions, and Structural Vector Autoregressions in Macroeconomics". In: John B. Taylor and Harald Uhlig (eds.), *Handbook of Macroeconomics*, edition 1, volume 2A, chapter 8, pages 415–525, Elsevier.

[83] Stock, James H., and Mark W. Watson. (2017) "Twenty Years of Time Series Econometrics in Ten Pictures", *Journal of Economic Perspectives*, 31(2): 59–86.

[84] Sutskever, Ilya, James Martens, George Dahl, Geoffrey Hinton. (2013) "On the Importance of Initialization and Momentum in Deep Learning", Proceedings of the 30th International Conference on Machine Learning,

[85] Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich. (2014) "Going deeper with convolutions", arXiv:1409.4842.

[86] Theil, Henri. (1954) *Linear aggregation of economic relations*, Amsterdam: North-Holland Publishing Company.

[87] Uhlig, Harald. (2005)"What Are the Effects of Monetary Policy on Output? Results from an Agnostic Identification Procedure", *Journal of Monetary Economics*, 52: 381–419.

[88] Uribe, Martín. (2018) "The Neo-Fisher Effect: Econometric Evidence from Empirical and Optimizing Models", Manuscript.

[89] Vlachas, Pantelis R., Wonmin Byeon, Zhong Y.Wan, Themistoklis P. Sapsis and Petros Koumoutsakos. (2018) "Data-driven Forecasting of High-dimensional Chaotic Systems with Long Short-term Memory Networks", *Proceedings of the Royal Society A*, 474: 20170844.

[90] Wager, Stefan, Sida Wang, and Percy S Liang. (2013) "Dropout Training as Adaptive Regularization". In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, 26: 351–359.

[91] Welling, Max, and Yee Whye Teh. (2011) "Bayesian Learning via Stochastic Gradient Langevin Dynamics", Proceedings of the 28th International Conference on Machine Learning.

[92] Williamson, Stephen. (2016) "Neo-Fisherism: A Radical Idea, or the Most Obvious Solution to the Low-Inflation Problem?", Federal Reserve Bank of St. Louis Regional Economist, July: 5–9.

[93] Wilson, D. Randall, and Tony R. Martinez. (2003) "The General Inefficiency of Batch Training for Gradient Descent Learning", *Neural Networks*, 16: 1429–1451.

[94] Zaremba, Wojciech, Ilya Sutskever, Oriol Vinyals. (2015) "Recurrent Neural Nerwork Regularization", arXiv:1409.2329.

[95] Zellner, Arnold. (1962) "An Efficient Method of Estimating Seemingly Unrelated Regressions and Tests for Aggregation Bias", *Journal of the American Statistical Association*, 57(298): 348–368.

[96] Zellner, Arnold, and Henri Theil. (1962) "Three-Stage Least Squares: Simultaneous Estimation of Simultaneous Equations", *Econometrica*, 30: 54–78.

[97] Zhang, Chiyuan, Samy Bengio, Moritz Hardt, Benjamin Recht, Oriol Vinyals. (2017) "Understanding Deep Learning Requires Rethinking Generalization", arXiv:1611.03530.