WARWICK
ECONOMICS

Predicting Inflation with Neural Networks

Livia Paranhos

April 2021 No: 1344

Warwick Economics Research Papers

ISSN 2059-4283 (online)
ISSN 0083-7350 (print)

Predicting Inflation with Neural Networks

Livia Paranhos*

University of Warwick

First version: November 2020

This version: March 2021

Abstract

This paper applies neural network models to forecast inflation. The use of a particular recurrent neural network, the long-short term memory model, or LSTM, that summarizes macroeconomic information into common components is a major contribution of the paper. Results from an exercise with US data indicate that the estimated neural nets usually present better forecasting performance than standard benchmarks, especially at long horizons. The LSTM in particular is found to outperform the traditional feed-forward network at long horizons, suggesting an advantage of the recurrent model in capturing the long-term trend of inflation. This finding can be rationalized by the so called long memory of the LSTM that incorporates relatively old information in the forecast as long as accuracy is improved, while economizing in the number of estimated parameters. Interestingly, the neural nets containing macroeconomic information capture well the features of inflation during and after the Great Recession, possibly indicating a role for nonlinearities and macro information in this episode. The estimated common components used in the forecast seem able to capture the business cycle dynamics, as well as information on prices.

Keywords: forecasting, inflation, neural networks, deep learning, LSTM model

*PhD Candidate, Department of Economics, University of Warwick, Coventry, West Midlands, CV4 7AL, UK. E-mail: L.Silva-Paranhos@warwick.ac.uk

1 Introduction

A good forecast model for inflation is essential for economic agents and policy makers. Yet, inflation dynamics has evolved substantially in the past decades, and the search for a reliable model is still an ongoing research question. Up until recently, it was hard to improve over simple univariate models for inflation, such as the unobserved components with stochastic volatility (UC-SV) model of Stock and Watson (2007), or autoregressive models. But the recent advances in machine learning methods have spurred the interest of economists in different techniques for inflation prediction, yielding promising results so far.[1]

This paper investigates the ability of neural networks to forecast inflation, with a special focus on recurrent neural networks. On the one hand, neural networks are well known by their flexibility in modelling nonlinearities while able to process large amounts of data efficiently.[2] On the other, recurrent neural networks, a sub-class of neural networks, were specifically designed to model sequences of observations, and are therefore appealing for the study of time series. I consider a specific recurrent neural network to predict inflation, the long-short term memory (LSTM) model. There are two main reasons why this model is well suited for this task.

The first is related to the way the model handles past information. While the traditional feed-forward neural network processes all input lags forwards and simultaneously in the network, recurrent models process each time period sequentially, where the input of a given time step is the output of the previous one. This recursion usually continues until a fixed lag L is reached. In practice, this translates into modelling explicitly the dependence between consecutive time periods along the series, which may explain the wide success of the LSTM model in the fields of speech recognition (audio to text), text translation (text to text), and sentiment analysis (text to rating), to name a few.

The second reason that makes the LSTM model attractive refers to its so called "long memory". This feature differentiates the LSTM from the plain recurrent neural network and refers to its ability in using information far in the past as long as this helps improving the accuracy of the prediction. In practice, this is achieved by incorporating in the recurrent model a number of filters that control the flow of information across time. This long-memory characteristic may be particularly important to predict the long-term trend of the series. Intuitively, it is likely that in

[1] Medeiros et al. (2019) is an important example, where the authors find that random forests significantly improve upon standard benchmarks when predicting US inflation.

[2] The universality theorem of neural networks, first studied by Cybenko (1989), suggests that a relatively simple feed-forward model can approximate any continuous function up to an arbitrary degree of accuracy.

order to predict far away in the future the practitioner is better off if she has a good understanding of the dependence of observations across a relatively long period of time. My results indeed indicate that recurrent networks overperform the feed-forward model (as well as commonly used benchmarks) in terms of out-of-sample performance at the one and two years ahead horizons.

Although the focus of the analysis resides on the recurrent model, other neural network structures are also considered for comparison. These are a feed-forward neural network and a combination of the LSTM and the feed-forward. Broadly speaking, the class of neural network models itself is very appealing for economic forecasting, and inflation forecasting in particular, given its ability in modelling highly nonlinear processes as well as in providing a solution to the curse of dimensionality.

The study of nonlinearities is important to inform macroeconomic theory as well as for economic forecasting (Barnett et al., 2015 is an early contribution on the topic). When it comes to inflation specifically, there is still a vivid, on-going debate on whether the slope of the Phillips curve would be time-varying (Stock and Watson, 2019, Hazell et al., 2020). Additionally, the inflation forecasting literature points to considerable out-of-sample gains of nonlinear machine learning methods over linear specifications (e.g. Nakamura, 2005, Sermpinis et al., 2014, Medeiros et al., 2019; see next section for more details on the literature). This suggests that at least to some extent inflation behaves nonlinearly with respect to other macroeconomic variables, which supports the study of nonlinear models for inflation forecasting. Neural networks are well suited to model nonlinearities given their ability in approximating a wide range of nonlinear functions (Cybenko, 1989). This paper estimates relatively deep networks (the depth of the network is related to the number of stacked computational units), investigating the accuracy of networks with up to four layers of computational units.

The second advantage of using neural networks for the purpose of inflation prediction, or economic prediction more generally, is that it offers an effective solution to the curse of dimensionality. A common criticism of traditional models of inflation relates to the relative small amount of conditioning information used to forecast the inflation series, which may lead to an omitted-variable estimation bias (Stock and Watson, 2009 provide a survey). It is well understood however that more classical parametric methods (e.g. BVARs) are quickly overwhelmed by a large number of predictors because of the increased dimension of the parameter space. Data reduction methods represent a remedy to this problem, from which factor models are the classical example. The more recent deep learning techniques can be viewed as an alternative, nonlinear solution to

the curse of dimensionality.[3] These methods can produce relatively accurate predictions despite the large set of covariates. Intuitively, the estimation process of a neural network consists of finding the lower dimensional representation of the data by attributing negligible weights to irrelevant information, and relative higher weights to information that improves the prediction.

With the purpose of understanding the role of activity-related variables on inflation as part of the debate discussed in Stock and Watson (1999, 2009), I create distinct sets of predictors. Specifically, I consider three sets of inputs: a data set with inflation-only information, a pool of economic predictors excluding inflation data, and a combination of both. Importantly, the pool of economic predictors does not include inflation data as a way of removing the effect of inflation itself on the forecast. In this application, I consider 128 economic variables, extracted from the FRED-MD data base.[4]

The paper makes several empirical contributions. First, with respect to its accuracy, neural networks are found to beat common benchmarks mainly at medium-long horizons (most of the models are significantly superior to benchmarks at the two-year forecast horizon).[5] The LSTM model in particular is found to have better forecast performance than the feed-forward neural network, controlling for the same information set. In fact, as opposed to the feed-forward model, the number of parameters in the recurrent model does not increase with the number of lags, which facilitates the estimation with long series of past information. Second, I show that general macroeconomic information, as opposed to CPI data only, was important to forecast inflation during the Great Recession and in its aftermath, a result that is in line with other works in the literature suggesting that economic information plays a substantial role in the prediction during episodes of high uncertainty (Chakraborty and Joseph, 2017, Medeiros et al., 2019). Third, the output of the LSTM model provides interesting insights on the signals of the economy that are particularly important to predict inflation. These signals seem to capture the dynamics of the business cycle, as well as information on prices. Additionally, the set of economic predictors as implied by the estimation of the neural network models seems to be nonsparse, a result in line with the findings in Giannone et al. (2018) that support the view that dense models should be preferred in the context of economic forecasting, although variables on output, income and

[3] In fact, most of machine learning models are able to deal with a large set of covariates, deep learning models being a subset of this class.

[4] The FRED-MD data base is a significant compilation of monthly US data made available by McCracken and Ng (2016).

[5] The benchmarks are the autoregressive model of order 1, the UC-SV model of Stock and Watson (2007) and the factor-augmented distributed lag (FADL) model. Appendix D provides the specifications.

consumption are found to be important predictors for inflation.

The organization of the paper is as follows. The next subsection provides a brief review of the literature. Section 2 introduces the econometric framework and the neural network models. Section 3 presents the data, the out-of-sample performance results and further quantitative analysis. Appendices B and C provide details on the estimation procedure and model specification respectively. Section 4 concludes.

Related work

The traditional literature on inflation forecasting usually refers to Phillips curve-based models, in which inflation is a function of some activity-based variable and autoregressive terms. Although well-established, forecasts based on the Phillips curve have varying performance over time and can be quickly overperformed by univariate models, as the UC-SV model (Stock and Watson, 2007), the random walk model (Atkeson and Ohanian, 2001) or autoregressive models.[6] Other widely used benchmarks are BVARs (Giannone et al., 2015) and dynamic factor models (Stock and Watson, 2002, Ludvigson and Ng, 2007), the later being especially good at short horizons. However, forecasting inflation is far from an easy task, and despite the extensive literature, the search for a reliable model of inflation is still an open question.

The rapid improvement of machine learning techniques in recent decades have shifted the attention of econometricians to this class of models.[7] This paper contributes in particular to the inflation forecasting literature by studying the suitability of neural networks to predict the inflation series, and is more broadly connected to the growing literature of economic forecasting using machine learning tools.

So far, studies on inflation prediction using machine learning have shown very promising results and usually involve models describing nonlinear mappings between inputs to outputs, such as random forests (Medeiros et al., 2019) and support vector regressions (Sermpinis et al., 2014). Although there is now a substantial number of papers that addresses the performance of neural networks in the context of inflation forecasting, most of them adopts the basic algorithm of

[6] Hasenzagl et al. (2018) are an exception, and find that a semi-structural model of inflation featuring a Phillips curve framework provides a good forecast performance for CPI inflation, especially at long horizons.

[7] Coulombe et al. (2020) provide a general framework to assess the usefulness of machine learning models to macroeconomic forecasting. In the field of economics more broadly, see for example Mullainathan and Spiess (2017) and Athey (2019) for an assessment of these methods applied to policy analysis and causal inference respectively. For financial applications, see e.g. Refenes and White (1998) and Gu et al. (2019). Varian (2014) provides a discussion about big data in economics.

the class, or the feed-forward neural network. Nakamura (2005) is an early implementation of a simple neural network for inflation prediction, and more recently Chakraborty and Joseph (2017) forecasts the two-year ahead inflation using a feed-forward neural network. I complement these studies by focusing on different neural network structures to forecast inflation, in particular the LSTM model which is new in this literature.[8] The main distinction between the recurrent model and the models previously studied is the assumption of dependence across time steps and involves explicitly modelling sequences of observations. This is specific of recurrent neural networks, while other models are silent about this time dependence. Another related work that applies the LSTM model in the context of economic forecasting is Cook and Hall (2017). The authors are interested in forecasting the civilian unemployment and find satisfactory results out-of-sample.

2 The framework

I consider two sets of predictive variables: $z_t = (z_{1t}, ..., z_{Nt})'$ generically denotes the macroeconomic variables used as predictors, and $w_t = (w_{1t}, ..., w_{Mt})'$ denotes the set collecting CPI inflation and its components, for $t = 1, ..., T$. Importantly, the set w_t is not contained in z_t, which allows me to isolate the predictive effect of the macroeconomic variables on inflation. Without loss of generality, I set the first element of w_t, i.e. w_{1t}, as the CPI inflation to be forecast, and denote it by y_t.

Let x_t be the set collecting the predictors at time t. In this paper, I consider predictor sets of the form $x_t = (z_t, ..., z_{t-(L-1)})'$, $x_t = (w_t, ..., w_{t-(L-1)})'$ or $x_t = (z_t, ..., z_{t-(L-1)}, w_t, ..., w_{t-(L-1)})'$, where the number of lags L may differ across sets.

I assume that the h-step ahead inflation y_{t+h} evolves nonlinearly with respect to the predictors x_t. Mathematically, y_{t+h} is modelled as a nonlinear function of the predictors, G, plus a non predictable component ε_t that is assumed to be *iid* with zero mean and variance σ^2 and independent of x_t,

$$y_{t+h} = G(x_t; \Theta_h) + \varepsilon_{t+h} \tag{1}$$

where Θ_h represents the model parameters. The underlying statistical problem consists therefore in estimating the unknown function $G : x_t \to y_{t+h}$.

In this application, G takes the form of a neural network structure, in which case fitting the

[8] Sermpinis et al. (2014) estimate a more traditional recurrent neural network (RNN) in the context of inflation prediction, while the present paper goes further and estimates the RNN with LSTM units, a more elaborate version of the RNN, able to model longer sequences of values.

function to the data corresponds to estimating Θ_h given a network architecture. An architecture $\mathcal{A}_G$ is specified as being a collection of choices that defines the functional form of G. It embeds two elements: the neural network model (or a combination of neural network models), and a set of parameters specific to each model, referred to as hyperparameters. Importantly, the choice of the network model is defined *ex-ante* by the researcher while the hyperparameters are optimized via grid search for each network model.[9] Making an analogy to the nonparametric literature, the hyperparameters can be viewed as tuning parameters that are model-specific, e.g. the bandwidth in kernel regression, or the choice of k in k-nearest-neighbors estimation. As neural network models, I consider the feed-forward (FF) neural network, the recurrent neural network with LSTM units, also referred here as the LSTM model for simplicity, and a combination of both, called the FF-LSTM model (the next sections provide details).

Let $\mathcal{S}_{\mathcal{A}_G}$ be the set of parameters specific to architecture $\mathcal{A}_G$. The parameters $\Theta_h \in \mathcal{S}_{\mathcal{A}_G}$ are estimated by minimizing the mean squared error loss

$$\hat{\Theta}_h = \underset{\Theta_h \in \mathcal{S}_{\mathcal{A}_G}}{argmin} \Big\{ \frac{1}{T-h} \sum_{t=1}^{T-h} \Big(y_{t+h} - G(x_t; \Theta_h) \Big)^2 \Big\} \tag{2}$$

for a given set of predictors x_t and target variable y_{t+h}, where the estimation is implemented by gradient descent (appendix B provides more details). The prediction $\hat{G}(x_t, \hat{\Theta}_h)$ can be interpreted as the conditional mean of the target variable.

The next sections present a detailed description of the three different model structures considered: the FF model, the LSTM model and the FF-LSTM model. They mainly differ from one another in two dimensions: they embed different neural network models, and they assume different predictor sets.

2.1 The feed-forward (FF) model

The feed-forward model is the fundamental structure of a neural network, inspired by the behavior of the human brain. The intuition behind this model is very simple. It consists of a potentially large number of simple elements, called nodes, that are organized into layers. The first is the input layer, carrying the input information, the last is the output layer, the one delivering the prediction of the model, and the layers in between are called hidden layers, in which the information flows in one direction, from inputs to outputs. A feed-forward network is said to be deep if it

[9] Details on the grid search process are provided in appendix C.

contains many hidden layers, normally ranging between 2 and 8 layers. Each node of the network, excluding the nodes in the input layer, processes the information coming in from the previous layer and delivers its output to the next layer. The computation behind these units consists on a weighted average over the output of all nodes in the previous layer, and a subsequent nonlinear transformation, referred to as the activation function. The weights applied to each node are the parameters of the model.

Formally, consider a feed-forward neural network with Q hidden layers. Let x_t be the predictor set, and $a_t^i \in \mathbb{R}^{n\times 1}$ be the hidden layer vectors containing n nodes each, for $i = 1, 2, ..., Q$.[10] The feed-forward model takes the form

$$G(x_t; \Theta_h) = g_{FF}(x_t) \tag{3}$$

where g_{FF} denotes a feed forward neural network with inputs x_t, and can be expressed as

$$\begin{aligned} g_{FF}(x_t) &= W_{Q+1} a_t^Q + b_{Q+1} \\ a_t^i &= ReLu(W_i a_t^{i-1} + b_i), \quad i = 1, 2, ..., Q \\ a_t^0 &= x_t \end{aligned} \tag{4}$$

where $\Theta_h = (\{W_i\}_{i=1}^Q, \{b_i\}_{i=1}^Q)'$ collects the parameters of the model. $\{W_i\}_{i=1}^Q$ are parameters relating the different layers of the network, $\{b_i\}_{i=1}^Q$ are intercept terms, and $ReLu : \mathbb{R} \to \mathbb{R}$ is the rectified linear unit activation function $ReLu(z) = max\{0, z\}$, applied element-wise. Other activation functions can also be considered, as the hyperbolic tangent and sigmoid functions, but the *ReLu* remains the preferred choice since it avoids estimation problems due to the simple form of its gradient. In fact, the *ReLu* function usually overperforms other activation functions in terms of statistical performance and computational cost.[11]

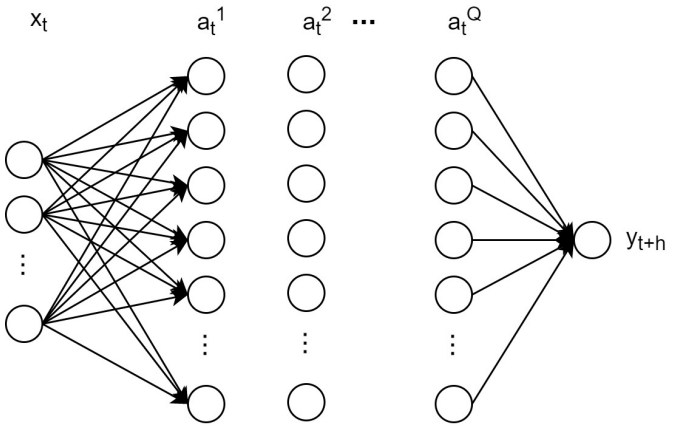
The hyperparameters of this model are the number of layers Q, the number of nodes per layer n and the number of lags L. Note here that the size of the input layer depends on the choice of the predictor set as well as the number of lags L (e.g. $(ML + 1) \times 1$, where the additional input stands for the bias term set equal to one), while the size of the output layer is fixed and set to one given the single-valued target variable. Figure 1 provides a graphical interpretation of this model.

[10] The restriction that each layer must contain the same n number of nodes is imposed for simplification purposes but could be relaxed, in which case the layer-specific number of nodes n would be selected via grid search.

[11] A recent work shows that the estimator of a deep *ReLu* network can achieve nearly optimal convergence rates for different constraints on the target function (Schmidt-Hieber, 2017).

Figure 1: The feed-forward neural network model

The figure is a representation of a feed-forward neural network with hidden layers a_t^i, for $i = 1, ..., Q$, a vector-valued input x_t, and a single-valued output y_{t+h}. The hidden layers are defined as $a_t^i = ReLu(W_i a_t^{i-1} + b_i)$ for $i = 1, 2, ..., Q$, with $a_t^0 = x_t$, while the target is modelled as a linear combination of the vector a_t^Q, $g_{FF} = W_{Q+1} a_t^Q + b_{Q+1}$. The arrows link the elements of the network and represent the parameters of the model, i.e. $(\{W_i\}_{i=1}^Q, \{b_i\}_{i=1}^Q)'$. The arrows in between the hidden layers as well as the intercept terms $\{b_i\}_{i=1}^Q$ are omitted for ease of visualization.



The feed-forward model, as the building block of a neural network, serves as a baseline for comparison with other, more complex neutral network structures. I consider estimating this model with both CPI-only data, $x_t = (w_t, ..., w_{t-(L-1)})'$, as well as with the pool of economic predictors excluding CPI data, $x_t = (z_t, ..., z_{t-(L-1)})'$. The first case, named FF-cpi, is a natural extension of the autoregressive model, in which a function relates lags of inflation (and its components) to the h-step ahead inflation through a (highly) nonlinear mapping. The total number of parameters in this model is $(ML + 1)n + (Q - 1)(n + 1)n + (n + 1)$. The second case, named FF-pool, forecasts inflation with a large set of macroeconomic predictors and its lags. This choice of input set is useful to identify the ability of other macroeconomic variables in predicting inflation. In this case, the total number of parameters amounts to $(NL + 1)n + (Q - 1)(n + 1)n + (n + 1)$.

2.2 The LSTM model

Recurrent neural networks (RNN) are promising models for time series forecasting, as they efficiently perform dimensional reduction while taking into account the time dependence within sequences of observations. This is possible because these models "remember" the information contained in previous time steps through a feedback loop. The main difference between the RNN

and the feed-forward network is the way the algorithm handles past information. While the latest processes all the input lags forwards and simultaneously in the network, the RNN processes each time step sequentially, allowing the output of a previous time step to be an input of the following one. This characteristic makes RNNs quite attractive to model time series dynamics.

As described formally below, RNNs have a so called *internal memory* that is updated at each time step. The model estimates the parameters such that its memory embeds the relevant information to forecast the target. Supposing that we feed the model with a large set of economic predictors, this internal memory would be interpreted as signals from the macroeconomic outlook that help predict inflation. This memory in practice takes the form of a vector that contain the cross-sectional and lagged information from the data set of predictors, and can be understood as common components in an analogy to factor analysis.

Consider the predictor set $x_t = (z_t, ..., z_{t-(L-1)})'$, with z_t being a N-vector of predictors. At each point in time the point forecast of the RNN is a function of its internal memory (or in the machine learning jargon, its hidden state). The internal memory is a p-vector denoted f_t, and is a function of the current input information z_t and the lagged internal memory f_{t-1}. Importantly, this recursion is limited to a fixed lag L in a way that past information can only be traced back up to lag L. In order to embed this idea in the notation, I write $f_{t|L} \equiv f_{t|t,t-1,...,t-(L-1)}$. This internal memory can be interpreted as a filter that reduces the dimension of the original predictor space N into a smaller number of common components p, where in general $p << N$, while incorporating past information through a recursion equation. The RNN model can be described as

$$G(x_t; \Theta_h) = g\Big(f_{t|L}\Big) \tag{5}$$

where g is a linear function on its inputs, and $f_{t|L}$ is the internal memory expressed as

$$\begin{aligned} f_{t|L} &= \Gamma(W' z_t + U f_{t-1|L} + b) \\ f_{0|L} &= 0 \end{aligned} \tag{6}$$

where Θ_h collects the parameters of $f_{t|L}$, $(W, U, b)'$, with $W \in \mathbb{R}^{N \times p}$, $U \in \mathbb{R}^{p \times p}$ and $b \in \mathbb{R}^{p \times 1}$, as well as those of the linear function g.[12] The hyperparameters of this model are the number of

[12] In this example, and throughout the paper, I assume an RNN with an architecture *many-to-one*, where at each time step the network receives the inputs, updates the internal memory, and only delivers the output after all lags $l = 1, ..., L$ have been processed through the recursion equation. Other variations exist, such as the specification *many-to-many*, in which the network delivers an output at each time step. See Goodfellow et al. (2016) for additional details on the different specifications of this model.

common components p in $f_{t|L}$ and the number of lags L. p is the equivalent of the number of nodes per layer n in the feed-forward network, and ultimately determines the degree of complexity of the model. Function Γ is applied element-wise, and may vary between applications, although the hyperbolic tangent and sigmoid functions are widely used in empirical applications.

The drawback of traditional RNNs is that they usually suffer from *vanishing gradients* that compromises the estimation process through a slow rate of improvement. Suppose that we seek to estimate an RNN with number of lags L. In brief terms, the estimation of the RNN involves computing the gradient of the loss function with respect to the parameters, which implies evaluating the gradient at every time step within sequences of observations of length L. If the parameters are significantly small (usually they are close to zero), the higher L, the smaller the contribution of observations sufficiently back in time, given the multiplicative effect of the chain rule and the fact that the derivative of the activation function is bounded by 1 (supposing the commonly used hyperbolic tangent or sigmoid functions). In other words, the model will not properly estimate long-term dependencies because the estimation process is compromised for sufficiently long sequences.[13]

The RNN with long-short term memory (LSTM) units solves this problem by avoiding the gradients to be too small, which is key to explaining the long-memory feature usually attributed to LSTMs. The intuition behind this algorithm relies on the existence of a *cell state* that turns out to be more stable than its counterpart in the traditional RNN, stabilizing the gradients as a consequence. This stability comes from the additive nature of the cell state, as well as the presence of filters that control the flow of information. These features together ensure suitable values for the gradient. For instance, if information from time step l shouldn't be forgotten to predict y_{t+h}, the parameters of specific filters are estimated accordingly so that the gradient at the l-time step is sufficiently large to account for this information when updating the model parameters. Intuitively, this mechanism allows the information to effectively flow across time periods.

It is common practice in machine learning to relate the internal memory of an LSTM to the target variable using a feed-forward network. This can be viewed as a generalization of the RNN case, where the internal memory is related to the prediction through the linear function g. Consider again the predictor set $x_t = (z_t, ..., z_{t-(L-1)})'$, with z_t being a N-vector of predictors. The

[13] In practice, the vanishing gradients problem may also occur in significantly deep feed-forward networks. The use of the *ReLu* activation function in these cases helps preventing the problem because its derivative is either 0 or 1.

LSTM model can be expressed as

$$G(x_t;\Theta_h) = g_{FF}\Big(f_{t|L}\Big) \tag{7}$$

where g_{FF} is the feed-forward network from equation 4. The internal memory $f_{t|L}$ of the LSTM takes a different, more complex format compared to the plain recurrent model introduced above. Its internal memory mainly reflects the LSTM as a control of the flow of information through time. Mathematically, $f_{t|L}$ is computed as

$$\begin{aligned}
f_{t|L} &= \phi_{t|L} \odot tanh(c_{t|L}) \\
c_{t|L} &= \psi_{t|L} \odot c_{t-1|L} + \zeta_{t|L} \odot tanh(W_c' z_t + U_c f_{t-1|L} + b_c) \\
\phi_{t|L} &= sigmoid(W_\phi' z_t + U_\phi f_{t-1|L} + b_\phi) \\
\psi_{t|L} &= sigmoid(W_\psi' z_t + U_\psi f_{t-1|L} + b_\psi) \\
\zeta_{t|L} &= sigmoid(W_\zeta' z_t + U_\zeta f_{t-1|L} + b_\zeta) \\
f_{0|L} &= 0, \quad c_{0|L} = 0
\end{aligned} \tag{8}$$

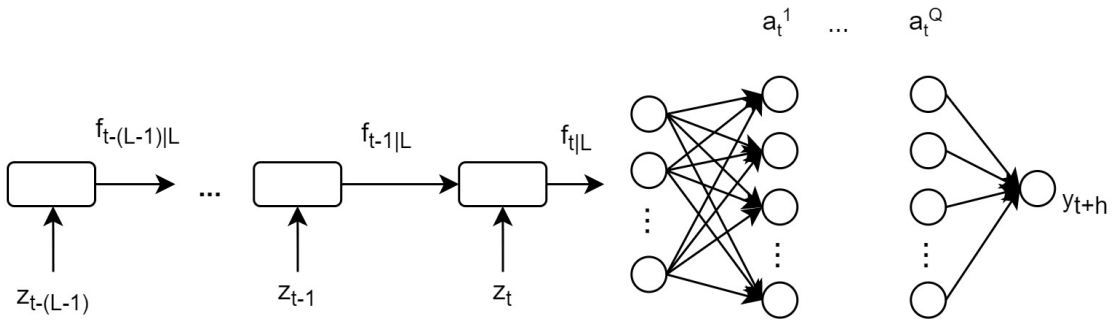
In this model, a cell state $c_{t|L}$ is updated recursively where a first filter denoted $\psi_{t|L}$ controls what *past* information to retain, and a second filter $\zeta_{t|L}$ controls what *new* information to retain at the current time period. The internal memory $f_{t|L}$ is then a function of the cell state where a final filter $\phi_{t|L}$ controls what information from the cell state to use for prediction.

Θ_h collects the parameters of $f_{t|L}$, $(W_{(j)}, U_{(j)}, b_{(j)})'$ for $j = c, \phi, \psi, \zeta$, where $W_{(j)} \in \mathbb{R}^{N\times p}$, $U_{(j)} \in \mathbb{R}^{p\times p}$, and $b_{(j)} \in \mathbb{R}^{p\times 1}$, as well as the parameters of the feed-forward network g_{FF}, $(\{W_i\}_{i=1}^{Q}, \{b_i\}_{i=1}^{Q})'$, where W_i and b_i have appropriate dimensions given n and Q. The functions sigmoid and hyperbolic tangent (tanh) are applied element-wise, and the symbol $\odot$ is the element-wise multiplication of two vectors. In this setting, the hyperparameters are the number of common components p in $f_{t|L}$, the number of lags L, the number of nodes per layer n in the feed-forward network as well as the number of layers Q. A graphical representation of the model is provided in figure 2.

I consider estimating this model with two data sets: the pool of economic predictors excluding CPI data, $x_t = (z_t, ..., z_{t-(L-1)})'$, and the full data set, $x_t = (z_t, ..., z_{t-(L-1)}, w_t, ..., w_{t-(L-1)})'$. The first case, called LSTM-pool has a total number of parameters of $4(Np + p^2 + p) + (p+1)n + (Q-1)(n+1)n + (n+1)$, while the second case, called LSTM-all, has a total number of parameters of

Figure 2: The LSTM model

The figure represents an LSTM model that receives the predictor set $x_t = (z_t, ..., z_{t-(L-1)})'$. On the left, the LSTM structure is shown unfolded, up to L lags. The output of the LSTM $f_{t|L}$ in turn enters as the predictor set of a feed-forward network, depicted on the right, which predicts the single-valued output y_{t+h}.



$4((N+M)p + p^2 + p) + (p+1)n + (Q-1)(n+1)n + (n+1)$.

2.3 The FF-LSTM model

The third model structure analysed in this paper, called FF-LSTM, is constructed based on the two models described above. It can be viewed as an augmented FF-cpi model, where CPI data $x_t = (w_t, ..., w_{t-(L-1)})'$ is combined to information on the state of the economy to form a composite input to a feed-forward neural network. The state of the economy is taken to be the p-dimensional internal memory $f_{t|L}$ of an LSTM that receives information on the pool of predictors excluding CPI data, $x_t = (z_t, ..., z_{t-(L-1)})'$. For clarification purposes, I rename the predictor sets as $x_t^w = (w_t, ..., w_{t-(L-1)})'$ and $x_t^z = (z_t, ..., z_{t-(L-1)})'$, and write $f_{t|L}(x_t^z)$.

In fact, this model can be interpreted as a generalized dynamic factor model, in which inflation is a (usually linear) function of common components and its lags, describing the economic state, as well as lags of inflation itself. A dynamic factor model is estimated as a benchmark (the FADL model specified in appendix D) and receives the same predictor set for the estimation of the factors as the LSTM model, or $x_t = (z_t, ..., z_{t-(L-1)})'$.

The FF-LSTM is therefore defined as a feed-forward model g_{FF} that receives the composite predictor set $(x_t^w, f_{t|L}(x_t^z))'$, which is essentially an augmented FF-cpi with additional input information on the common components $f_{t|L}(x_t^z)$. Mathematically,

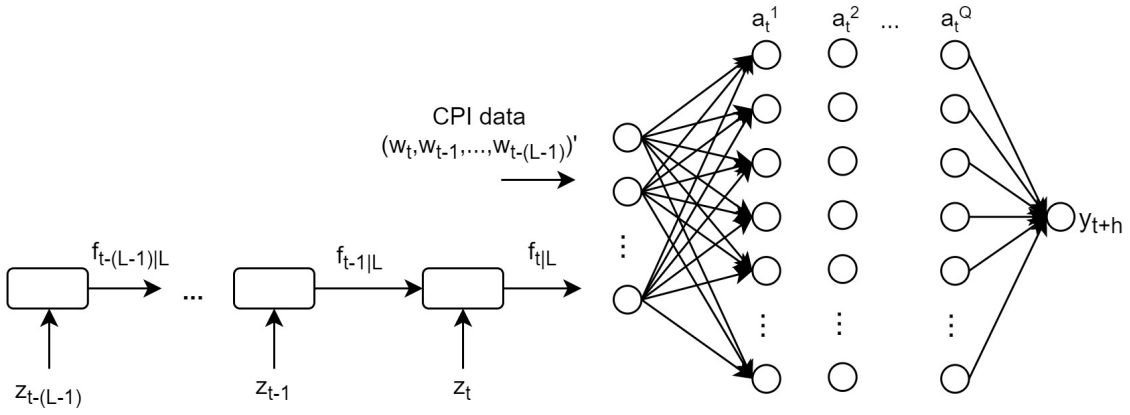
$$G(x_t; \Theta_h) = g_{FF}\Big((x_t^w, f_{t|L}(x_t^z))'\Big) \tag{9}$$

where g_{FF} is the feed forward network from equation 4, $f_{t|L}$ takes the same form as in equation

8, and Θ_h collects the parameters of $f_{t|L}$ as well as the feed-forward network g_{FF}. Similarly to the LSTM model, the hyperparameters are the number of common components p in $f_{t|L}$, the number of lags L, the number of nodes per layer n in the feed-forward network as well as the number of layers Q. In this setting, there are two distinct L lags to be determined, each specific to a set of predictors, x_t^z and x_t^w. The number of parameters in this case amounts to $4(Np + p^2 + p) + (p + ML + 1)n + (Q - 1)(n + 1)n + (n + 1)$. Figure 3 provides a graphical illustration.

Figure 3: The FF-LSTM model

The FF-LSTM model is essentially a feed-forward network with a composite predictor set: (i) data on the CPI and components $(w_t, ..., w_{t-(L-1)})'$, and (ii) a p-dimensional internal memory $f_{t|L}$ of an LSTM structure computed from the pool of economic predictors $(z_t, ..., z_{t-(L-1)})'$. In this way, the FF-LSTM model nests both the FF-cpi and LSTM-pool models.



3 Empirical Analysis

3.1 Data

The data used in the empirical study is collected from the FRED-MD data base, a compilation of monthly US data made available by McCracken and Ng (2016), and corresponds to the vintage of October 2019. This data set is comprised of 128 series with 730 observations each, spanning the period from January 1959 to October 2019. Table 2 in appendix A provides the description of all the series as well as information on the data transformation. Departing from this data set, I create two sets of variables: the first comprises the ten series with direct CPI information, i.e. the CPI and its components (corresponding to series indexed by 110 to 199 in table 2), and the second

comprises all the remaining series. The first set is denoted w_t, and the second, z_t.

The series undergo a sequence of transformations before estimation. First, I transform the data following the specifications in McCracken and Ng (2016) to guarantee stationarity. The only exception is the CPI series, which here is specified in log differences, $\pi_t = log(P_t) - log(P_{t-1})$, where P_t is the price index at time t. Second, missing observations are replaced with the unconditional mean for each series. And third, the data is normalized within the interval $[-1, 1]$. This normalization is carried out in the in-sample set and extrapolated to the out-of-sample set such that there is no look-ahead bias. It is worth mentioning that neural networks may very well handle non-stationary data, hence the transformations that guarantee stationarity could potentially be relaxed. However it is much less evident whether these models can perform well in the presence of covariates with numerical values significantly different from one another. In fact, the evidence suggests that neural networks perform better when the inputs share a similar order of magnitude.[14] It is therefore very common in the literature to implement feature scaling to the data (feature standing for covariates), which is similar to assuming an equal importance of covariates *ex-ante*.

3.2 Predictions as an ensemble

The usual non-convex loss function of neural networks implies that the estimated parameters are in general very sensitive to initial values (initial parameters are randomly drawn from a specific uniform distribution; see appendix B). In practice, this means that the neural network predictions will be very much dependent on the initialization. To overcome this issue, the empirical literature usually adopts the solution of averaging out the predictions from models estimated with different initial values.

Consider the set of models $\{\mathcal{M}_k\}_{k=1}^{K}$ estimated from the same neural network architecture but with different initial values. Let $\{\hat{y}_{k,t+h}\}_{t=1}^{P}$ be the forecast associated with model $\mathcal{M}_k$, where P is the out-of-sample size. The ensemble prediction at period $t+h$ is defined as $\hat{y}_{ens,t+h} = \frac{1}{K}\sum_{k=1}^{K}\hat{y}_{k,t+h}$, which is essentially a model-averaging technique that attributes an equal weight to each forecast $\hat{y}_{k,t+h}$. Section 3.6 shows empirically that this solution is at least as good as the individual forecasts in terms of mean squared error according to a standard Diebold and Mariano (1995) (DM) test procedure.

[14] This fact is related to the gradient descent optimization process, which seems to converge much faster when the input data is normalized (see Ioffe and Szegedy, 2015). For details on the optimization procedure, see appendix B.

I therefore adopt this ensemble technique and estimate each neural network specification $K = 1400$ times letting vary the initialization.[15] The main out-of-sample results reported below (section 3.5) refer to the ensemble prediction $\hat{y}_{ens,t+h}$ across the 1400 forecasts.

3.3 The dynamics of the internal memory $f_{t|L}$

In this section, I extract the internal memory $f_{t|L}$ of estimated neural network models and evaluate it over the full sample period as an attempt to characterize its dynamics and association with the business cycle. The focus of this exercise is therefore on the three models embedding LSTM structures, i.e. LSTM-pool, LSTM-all and FF-LSTM (the optimal architecture for each model is selected *ex-ante* via grid search; see table 3 in appendix C).

As previously discussed, the p-dimensional internal memory $f_{t|L}$ of the LSTM is estimated from a large set of economic predictors, and can be interpreted as signals of the economy that are important to predict inflation. The elements of $f_{t|L}$ can be similarly interpreted as common components of variation from the set of predictors. However, the vector $f_{t|L}$ is not uniquely determined given the common lack of identification of neural network models. This is usually caused by the symmetric nature of neural networks (such that a swap of nodes from a given layer leads to the same function value) as well as the cross-dependence of parameters inside the network. Appendix B provides a more detailed discussion on that matter.

In order to recover a single vector $f_{t|L}$, I first estimate a set of models $\{\mathcal{M}_k\}_{k=1}^{K}$ with different initial values but same neural network architecture. I then select the model $\mathcal{M}_k$ with lowest out-of-sample error over a validation sample (see appendix C), which essentially implies selecting an optimal initialization.[16] Finally, given a model $\mathcal{M}_k$, it is possible to evaluate the internal memory $f_{t|L}$ over the full sample period using rolling windows of L observations, for $t = L, ..., (T - h)$.

The outcome of this exercise is displayed in figure 4 for the three models of interest at horizon $h = 24$. The figure plots the p components of the internal memory $f_{t|L}$ together with the CPI inflation series computed at the annual rate of change. Note that the dimension p is selected via grid search, where $p = 2$ for all three models. Also note that the internal memory $f_{t|L}$ is constrained to lie in the interval $[-1, 1]$, as imposed by the structure of the LSTM.[17]

[15] In order to guarantee optimal computational time, the number of iterations was chosen such that it is proportional to the number of available processors.

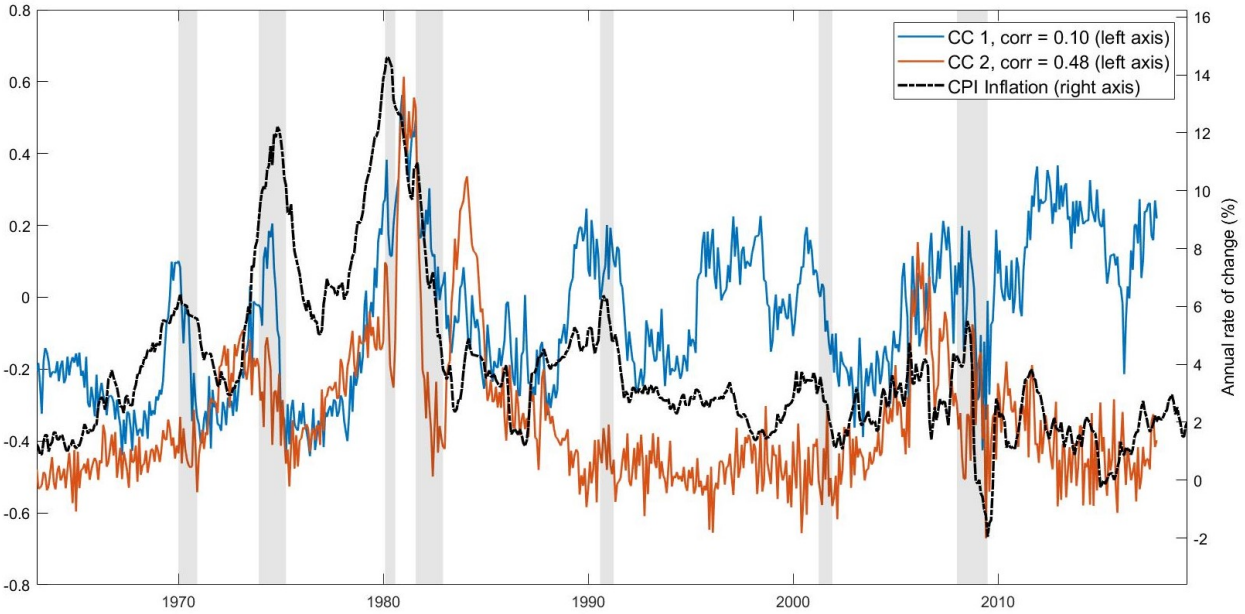
[16] Note that the ensemble strategy cannot be applied here given the non-identifiability of the model. Indeed, it would be meaningless to average out non-identifiable components.

[17] The vector-valued output of the LSTM takes the form of an element-wise multiplication between a sigmoid function, constrained to $[0, 1]$, and a hyperbolic tangent function, constrained to $[-1, 1]$ (see equation 8).

Figure 4: The p-components of the internal memory $f_{t|L}$

The figure plots the $p = 2$ components of the internal memory $f_{t|L}$ of three estimated neural network models together with the CPI inflation series. The plots correspond to estimations at horizon 24 and cover the full sample period. The correlation between each component and the inflation series is indicated in the legend. The internal memory $f_{t|L}$ can be interpreted as signals of the economy that are relevant to predict inflation, where the number of components is pre-selected via grid search. The grey areas are NBER recessions.

(a) LSTM-pool



(b) LSTM-all

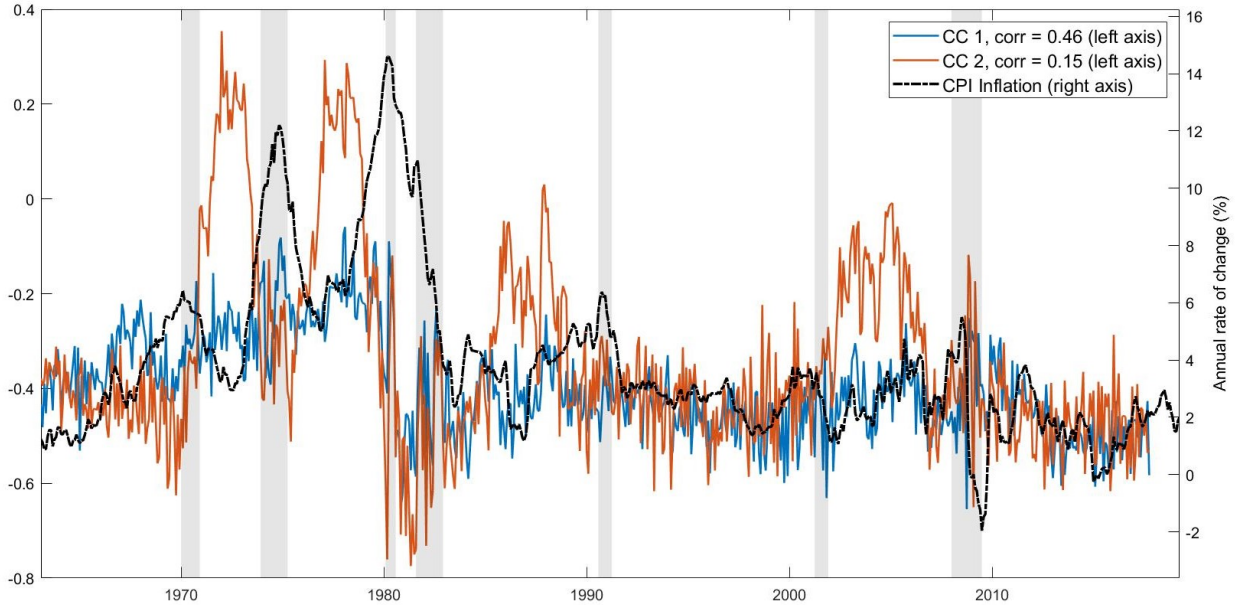
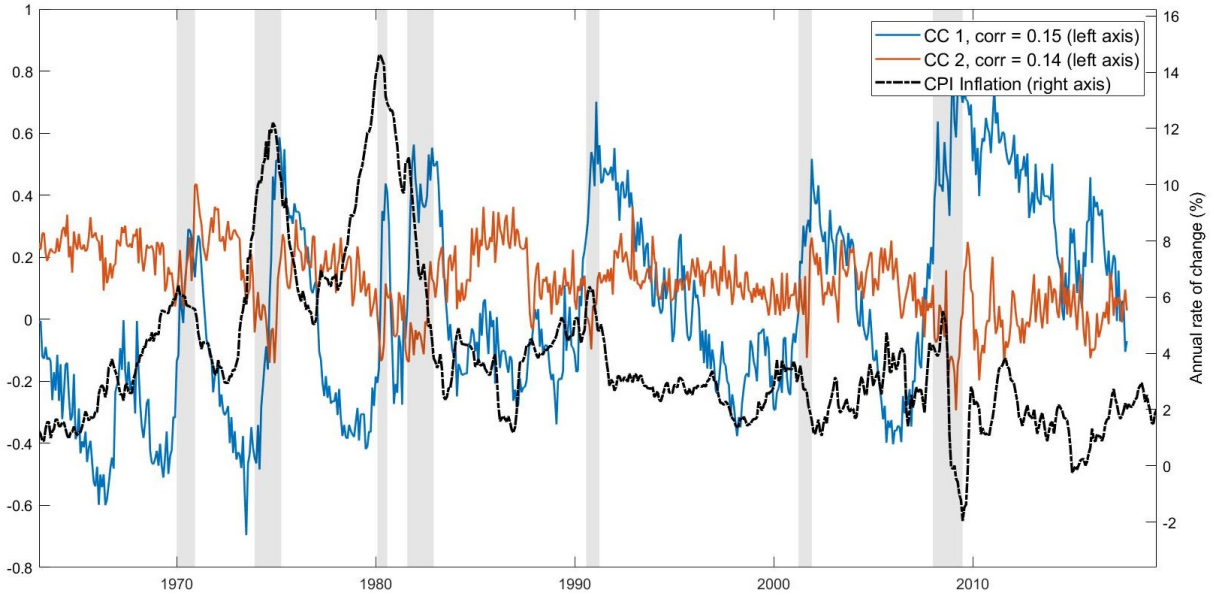


Figure 4: Common components - *continued*

(c) FF-LSTM



I discuss two points. First, the components of the internal memory $f_{t|L}$ seem to capture well some elements of the inflation series. In particular, some of the series track well the hyperinflation period during the 1970's and 1980's, as well as the downward trend of inflation during the last decades. This is mostly visible for the LSTM-pool and LSTM-all models. Note also that some of the series are successful in capturing the abrupt movements of inflation during the Great Recession (LSTM-pool and FF-LSTM).

Second, the internal memory $f_{t|L}$ also conveys information on the business cycle. This is more evident in the FF-LSTM model, where one of the series fluctuates significantly around recessions. Note that for models where the internal memory $f_{t|L}$ is the only element used for prediction (i.e. LSTM-pool and LSTM-all), one of the $f_{t|L}$ components shows significant correlation with the inflation series, while this is not the case for the FF-LSTM model for which the predictor set is augmented with CPI data. Instead, the internal memory $f_{t|L}$ of this later model appears to be capturing only the information in *excess* of what is already provided by the CPI data. In this case the components of $f_{t|L}$ can be viewed as filtered versions of previous models' $f_{t|L}$s, where the contribution of CPI data for prediction is removed. This eventually explains the explicit relation of $f_{t|L}$ to the business cycle according to the estimation of the FF-LSTM model.

3.4 Variable selection

In this section, I provide insights on variable selection as implied by the estimation of the models containing either the pool of economic predictors (FF-pool, LSTM-pool) or all the variables in the data set (LSTM-all and FF-LSTM).

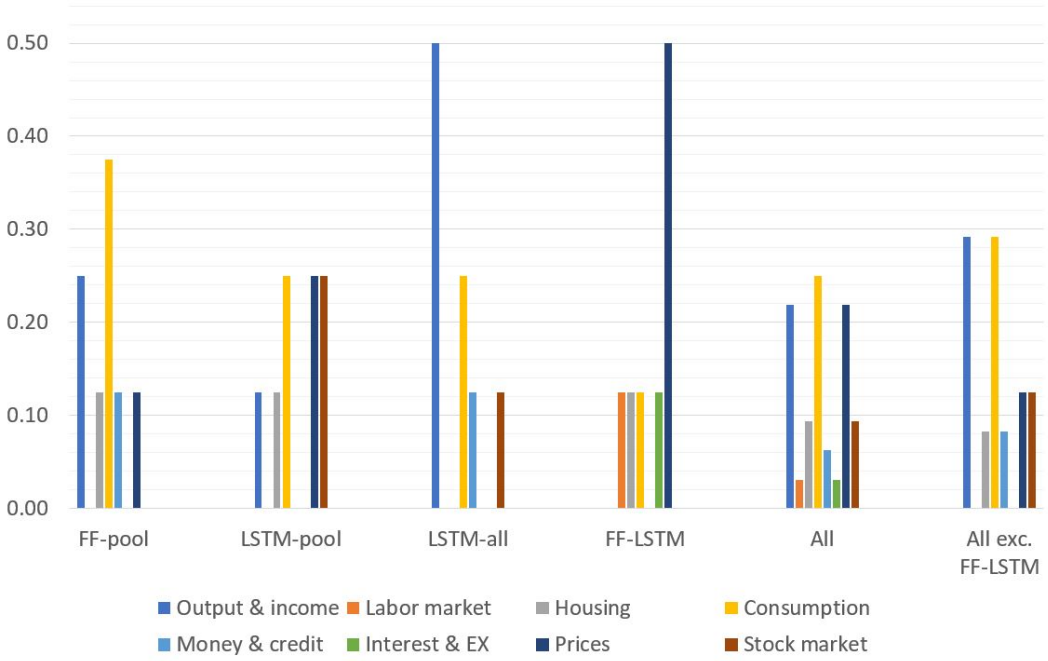
The analysis employs a perturbation method that identifies the importance of each variable to the prediction. Specifically, I re-estimate the models such that at each estimation a specific input variable i is perturbed with a 3 standard deviation shock. The resulting prediction series is then compared to a benchmark series where none of the variables are shocked using a mean squared error-type loss over the out-of-sample period, which is referred as the gain of variable i. The variable-specific gains are then averaged out within groups, where I follow McCracken and Ng (2016) and allocate the variables into eight groups: output and income, labor market, housing, consumption, money and credit, interest rates and exchange rates, prices and stock market.[18]

Figure 5 provides a visual outlook of the gains across groups and models. The values displayed correspond to the frequency that each group had the highest gain and/or the second highest gain across horizons (3, 6, 12 and 24). Results are presented across different levels of aggregation: at the model level, over all models, and over all models excluding the FF-LSTM. I discuss two points. First, the groups "output and income" and "consumption" clearly stand out in variable importance. When looking at frequencies across all models, "consumption" presents the largest frequency of high gains, followed by "output and income" and "prices". Groups as "labour market" and "interest and exchange rates" appear to have little relative importance to the forecast. Note that the large frequency of the "price" group implied by the FF-LSTM is compatible with its structure. Recall that in this model CPI data enters directly in the feed-forward part of the model, rather than through the LSTM, such that a shock to a CPI component is likely to have a larger impact on the prediction than a variable processed first by the LSTM. Excluding the FF-LSTM when aggregating gains across models, "output and income" and "consumption" present again the largest frequency of high gains. And second, including CPI data in the model does not increase the relative importance of the price group (LSTM-pool *vs* LSTM-all). This means that CPI data itself does not seem crucial when already controlling for macroeconomic information. This result supports the findings of Giannone et al. (2018) that promote the use of dense models in the context of economic forecasting.

[18]For models that do not include data on the CPI and its components (i.e. FF-pool and LSTM-pool), the group "Prices" does not include these variables.

Figure 5: Variable selection

The values displayed correspond to the frequency that each group had the highest gain and/or the second highest gain across horizons (3, 6, 12 and 24). I refer to the gain of variable i as the mean squared error over the out-of-sample period between a benchmark prediction and a prediction resulting from perturbing variable i by a 3 standard deviation shock. Variable-specific gains are then averaged out within economic-based groups. Results are presented across different levels of aggregation: at the model level, over all models, and over all models excluding the FF-LSTM.



3.5 Out-of-sample analysis

In this section, the performance of the neural network models is compared to three benchmarks, the autoregressive model of order 1, the UC-SV model of Stock and Watson (2007), and the factor-augmented distributed lag (FADL) model. These are widely used benchmarks in the literature of inflation forecasting and are presented in appendix D, together with details on their estimation.

Table 1 presents the out-of-sample results. The entries correspond to loss ratios with respect to the AR(1) benchmark over the out-of-sample period (2006M08 - 2019M10), where both the root mean squared error and the mean absolute error losses are considered. The results indicate that the neural network models have in general a lower forecast error than the benchmarks, especially at the one and two years ahead forecast horizons. However, statistically significant results according to a DM test are mainly found at the two year horizon, implying that these models are good at forecasting the long term trend of inflation, but are less appropriate for shorter horizons. The models incorporating LSTM structures do particularly well compared to the

feed-forward specifications. For instance, keeping constant the predictor set, the LSTM-pool is at least as good as the FF-pool across almost all horizons (except at the 6-step ahead). Regarding the importance of predictors for the forecast, it is possible to infer that the use of CPI data does not necessarily imply in greater forecast performance once controlling for a large set of macroeconomic variables. This can be seen from the similar forecast accuracy of the LSTM-pool and LSTM-all models, the later incorporating CPI data on top of the pool of macroeconomic predictors. In the same way, it is possible to argue that the model carrying only CPI information, FF-cpi, does not excel compared to the alternatives, including those models without CPI data (i.e. FF-pool and LSTM-pool). Overall these results support the idea that the LSTM structure, when estimated in a big data environment, has advantages over commonly used benchmarks as well as over the feed-forward model.

It is important to note that a possible reason behind the poor performance of the FF-pool model could be linked to the large number of estimated parameters. For the same input set, the number of parameters of the FF-pool is almost 15 times the one of the LSTM-pool (see table 3 in appendix C). This difference is in part one of the advantages of the LSTM model with respect to the feed-forward structure: the LSTM is not penalized in terms of number of parameters as the number of lags increases. The intuition is straightforward given the recurrent nature of the LSTM, since all lagged values enter the model through the same recurrent cell, implying that the algorithm estimates only one set of parameters for all lags. On the other hand, the number of parameters of the feed-forward model is increasing in the number of lags, as each lag is treated as a new predictor.

As a way of illustrating the different model performances, figure 6 plots the realized inflation series together with the predictions of two neural network models, FF-pool and LSTM-pool, and the predictions of the factor model, at the 24-step ahead horizon. Note that the factor model is more sensitive to new information, and sees its performance penalized with respect to the neural networks once data on the financial crisis period comes in. The neural network models on the other hand are able to produce much smoother prediction series. In the comparison between the two neural network models, note that the FF-pool underestimates the trend of inflation for a relative long period, possibly indicating more unstable predictions given its large amount of parameters, as discussed previously.

In order to get a clearer picture of the models' performance over time, as opposed to the average performance, I employ the Fluctuation Test introduced by Giacomini and Rossi (2010).

Table 1: Out-of sample forecast performance

The table presents the loss ratios with respect to the AR(1) model for specific horizons over the period 2006M08 - 2019M10, as well as the data included in each model. In the column 'Data', 'CPI' is the CPI series (series 110 in table 2), 'CPI+cp.' is the CPI and its nine components (series 110 to 119 in table 2), 'Pool' groups all the series from table 2 except 'CPI+cp.', and 'All' refers to all series in table 2. The loss functions are the root mean squared error (RMSE) and the mean absolute error (MAE). Stars denote significance of the Diebold and Mariano (1995) test at a 10%, 5% and 1% level.

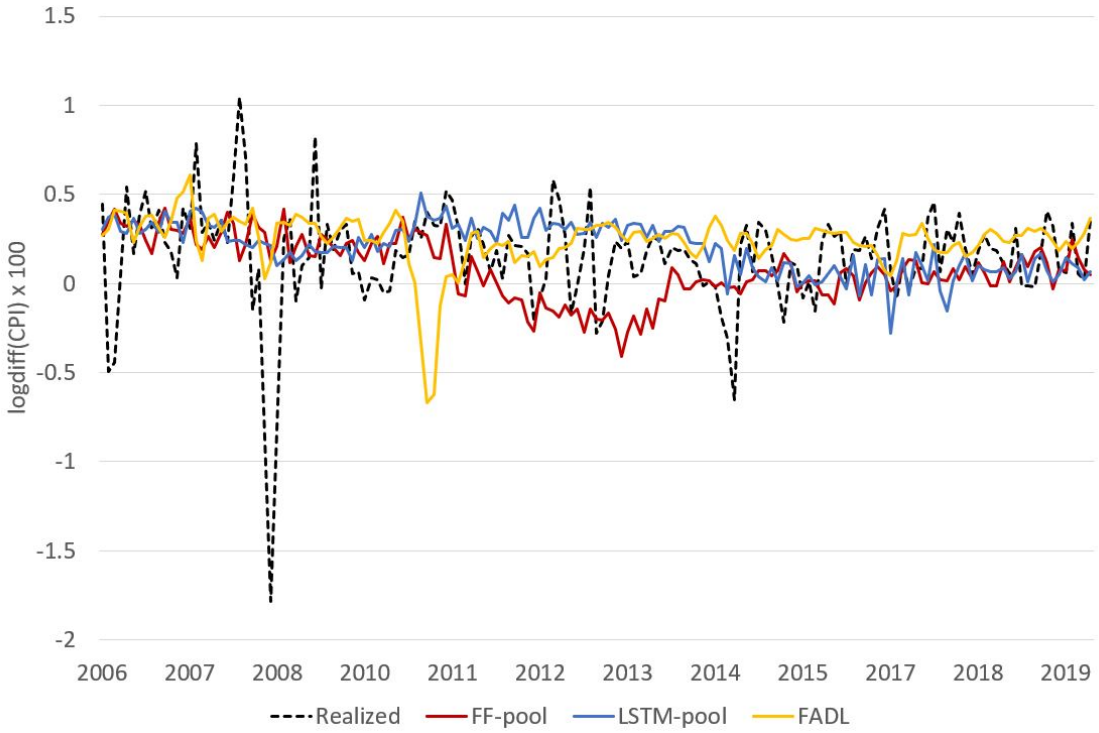
		Horizon (months)				
Model	Data	1	3	6	12	24
RMSE						
AR(1)	CPI	1.00	1.00	1.00	1.00	1.00
UC-SV	CPI	1.13	1.05	1.03	1.02	1.00
Factor-aug DL	Pool (excl CPI+cp.)	1.05	1.09	1.08	1.01	1.00
FF-cpi	CPI+comp	1.07	1.01	1.01	0.98	0.91**
FF-pool	Pool (excl CPI+cp.)	1.09	0.92	0.90*	0.94	0.99
LSTM-pool	Pool (excl CPI+cp.)	1.00	0.93	1.03	0.93*	0.92**
LSTM-all	All	0.98	0.94	1.04	0.92**	0.91**
FF-LSTM	All	1.06	0.99	0.99	0.97	0.89***
MAE						
AR(1)	CPI	1.00	1.00	1.00	1.00	1.00
UC-SV	CPI	1.05	1.05	1.04	1.00	1.08
Factor-aug DL	Pool (excl CPI+cp.)	0.99	1.07	1.07	0.98	1.01
FF-cpi	CPI+comp	1.03	0.98	1.00	0.94	0.91**
FF-pool	Pool (excl CPI+cp.)	1.09	0.95	0.95	0.98	1.02
LSTM-pool	Pool (excl CPI+cp.)	0.97	0.93	1.09	0.92	0.90**
LSTM-all	All	0.97	0.95	1.10	0.91	0.91**
FF-LSTM	All	1.02	0.96	0.97	0.91	0.88***

The proposed statistic tests for an equal forecast accuracy of two forecasting models and is robust to instability on their relative forecast performance. The statistic is much similar to the DM test procedure, except that it is computed over a rolling window of fixed size m. In this application, I use the one-sided test where the rejection of the null hypothesis (i.e. a test statistic larger than the critical value) implies that the candidate model is statistically superior to the benchmark for a given point in the out-of-sample period. The value of m is chosen such that $m/P \approx 0.3$, in which case the one-sided critical value at the 5% confidence level is 2.77 according to Giacomini and Rossi (2010).

Figure 7 provides the statistic series with respect to the AR(1) model. I discuss several points. First, I argue that models carrying broad macroeconomic information, as opposed to models

Figure 6: Realized *versus* predicted

This graph plots the realized inflation series (Realized) together with 24-step ahead predictions of two neural network models (FF-pool and LSTM-pool) and predictions of the factor-augmented model (FADL) over the out-of-sample period. Realized inflation is specified in log differences $\times 10^2$.

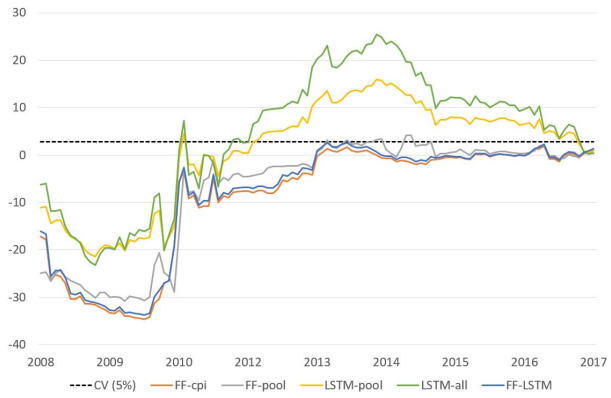


carrying only CPI information, are more appropriate to predict inflation, especially for certain periods during the business cycle. More specifically, there is a clear distinction between two groups of models during the period covering the Great Recession and its aftermath, i.e. from the end of 2008 until mid 2010. In this first part of the sample, the first group (FF-pool, LSTM-pool, LSTM-all) presents systematically a better forecast accuracy than the second group, or models that contain mainly CPI information (FF-cpi, FF-LSTM). Moreover the first group is statistically superior to the benchmark at the 5% level, while the second cannot beat the benchmark during the same period. The exception is at horizon 24 where all models have a similar, good performance. Still on the importance of macroeconomic information, note that the inclusion of CPI data in the LSTM model (LSTM-pool *versus* LSTM-all) does not increase significantly its performance, in fact we observe a decrease in performance over some periods of the sample (e.g. horizons 3 and 6). On the opposite side, the inclusion of the common components in the feed-forward model (FF-cpi *versus* FF-LSTM) proves to improve forecast performance for all horizons, supporting the

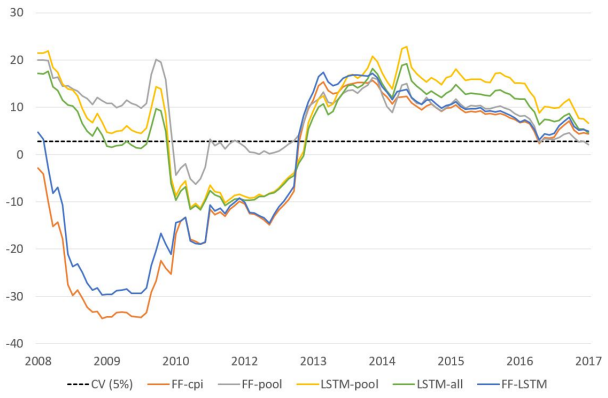
Figure 7: Test for equal forecast accuracy

The figure exhibits the one-sided fluctuation test (Giacomini and Rossi, 2010) for equal forecast accuracy between the neural network specifications and the $AR(1)$ benchmark. A positive statistic for a given point in time implies that the specified model has a lower forecast error than the benchmark over a window of 48 observations around that point. The dashed line represents the critical value at the 5% confidence level.

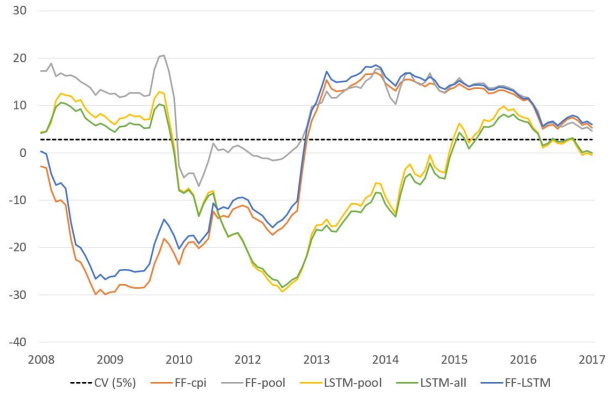
(a) Horizon 1



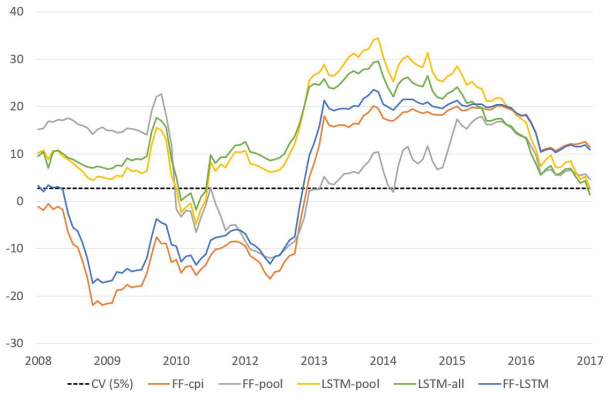
(b) Horizon 3



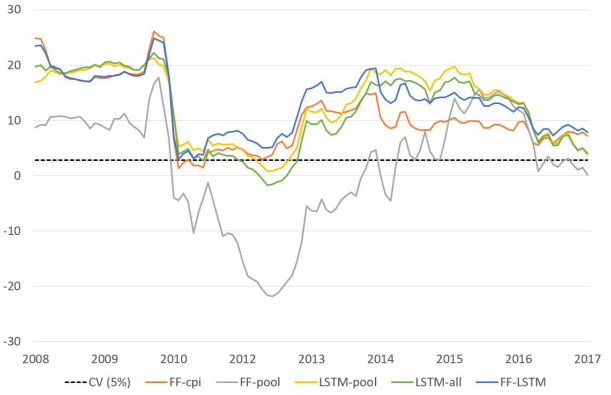
(c) Horizon 6



(d) Horizon 12



(e) Horizon 24



relevance of economic information also during normal times.

Second, the choice of the neural network model is crucial for forecast performance. Consider the models LSTM-all and FF-LSTM. Both receive the same input information but they treat the data in a slight different way. While the LSTM-all processes the entire data set within the LSTM structure, the FF-LSTM supplies the CPI information directly to the feed-forward network, which to some extent increases the weight of CPI data in the final prediction compared to other economic predictors. As figure 7 shows, the performance of these two structures differs substantially over time, and the larger importance given to CPI data under the FF-LSTM framework proved to be detrimental for its performance. This point substantiates the idea that the network architecture, leaving aside the estimation of the parameters, appears to play an important role in the forecasting exercise and should be considered with caution.

Third, the LSTM model shows a better forecast performance than the feed-forward network systematically over the out-of-sample period, especially at medium to long horizons. For a similar predictor set (FF-pool *versus* LSTM-pool), the only instances where the FF model outperforms the LSTM is during the unstable period of the crisis for horizons 3 and 12 as well as for the entire sample at horizon 6, where the LSTM shows a particularly bad performance. However if one considers horizons of policy interest (one and two years ahead), these results indicate that the LSTM model is an interesting candidate in light of its advantage with respect to usual benchmarks, but also with respect to other neural network structures. As previously discussed, this result is also robust to the data set of predictors, as the presence of CPI data in the input set does not affect the outcome significantly.

3.6 Uncertainty over initial values

As discussed before, neural network predictions are in general very sensitive to initial conditions. This is mainly a consequence of the non-convexity of the loss function, where the presence of non-linearities tends to complicate the search for a global optimum. In fact, it is likely that the algorithm will converge to some local optimum, which explains why predictions usually differ across estimations under different initial values.

Natural questions that emerge are how different are the performances of forecasts embedding different initializations, and more importantly how the performance of the ensemble forecast $\{\hat{y}_{ens,t+h}\}_{t=1}^{P}$ compares with the performance of individual forecasts $\{\hat{y}_{k,t+h}\}_{t=1}^{P}$, for $k = 1, ..., K$. To address these points, I compute a Diebold and Mariano (1995) test statistic of equal forecast

accuracy between the individual forecasts and the ensemble forecast for each $k = 1, ..., K$. This exercise essentially yields a distribution of the DM test statistic over specifications embedding different initial values.

Let $\{e_{k,t+h}\}_{t=1}^{P}$ and $\{e_{ens,t+h}\}_{t=1}^{P}$ be forecast errors associated with forecast $\{\hat{y}_{k,t+h}\}_{t=1}^{P}$ and with the ensemble forecast $\{\hat{y}_{ens,t+h}\}_{t=1}^{P}$ respectively over the out-of-sample period. Now consider a loss-differential series $\{d_{k,t+h}\}_{t=1}^{P}$, where $d_{k,t+h} \equiv [e_{k,t+h}^2 - e_{ens,t+h}^2]$, such that positive values are associated with a better performance of the ensemble prediction. For a given horizon h, I compute the following Diebold and Mariano (1995) test statistic of equal forecast accuracy:

$$\Delta_{k,h} = \frac{\bar{d}_{k,h}}{\sqrt{\frac{2\pi \hat{f}_d(0)}{P}}}, \qquad k = 1, ..., K \tag{10}$$

where $\bar{d}_{k,h} = \frac{1}{P}\sum_{t=1}^{P} d_{k,t+h}$ is the sample mean loss differential, and $\hat{f}_d(0)$ is the estimate of the spectral density of the loss differential at frequency 0, $f_d(0)$. A consistent estimate for $2\pi f_d(0)$ is obtained by taking a weighted sum of the sample autocovariances

$$2\pi \hat{f}_d(0) = \sum_{\tau=-(P-1)}^{P-1} 1\left(\frac{\tau}{S(P)}\right) \hat{\gamma}_d(\tau)$$

where $\hat{\gamma}_d(\tau) = \frac{1}{P}\sum_{t=|\tau|+1}^{P}(d_{k,t+h} - \bar{d}_{k,h})(d_{k,t+h-|\tau|} - \bar{d}_{k,h})$, and $1\left(\frac{\tau}{S(P)}\right)$ is the uniform lag window, taking the value of 1 if $\left|\frac{\tau}{S(P)}\right| \leq 1$ and 0 otherwise. $S(P)$ is the truncation lag that I define as $S(P) = P^{1/3}$ following standard practice.[19]

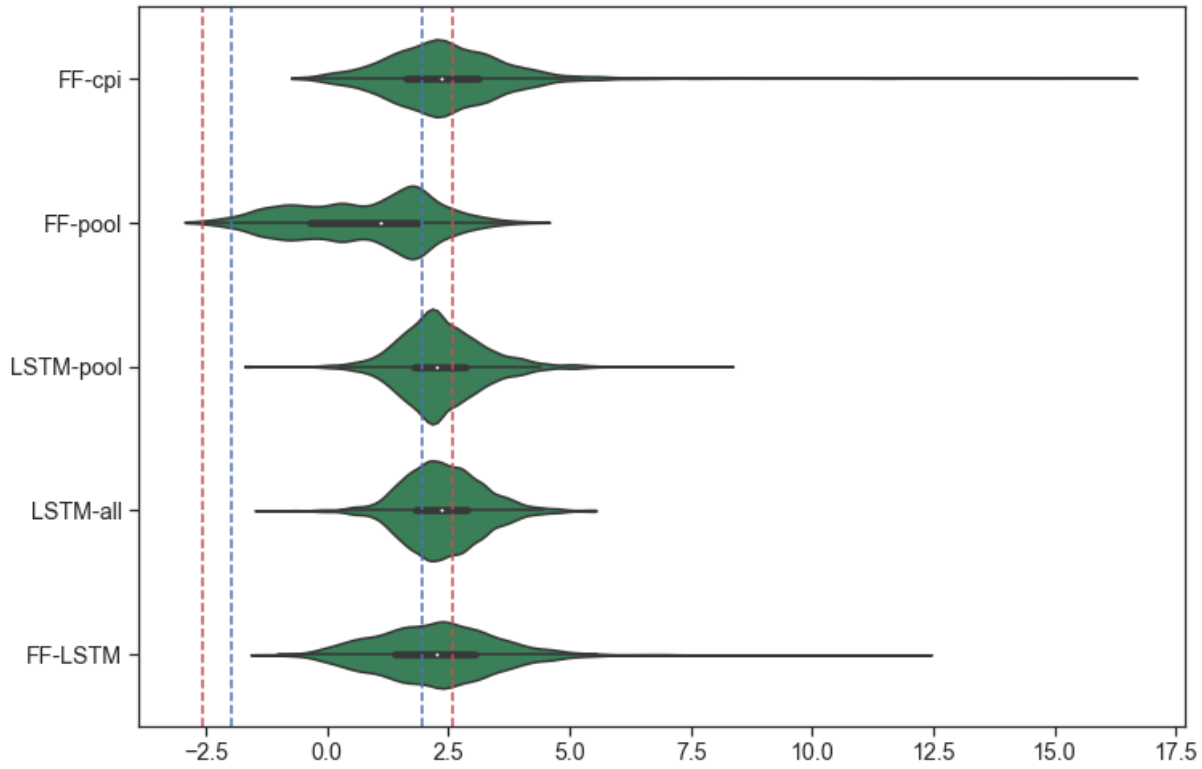
I carry out the analysis considering $h = 24$ and $K = 1400$. Figure 8 plots $\Delta_{k,24}$, for $k = 1, ..., 1400$, for each neural network model. Also plotted are the critical values at the 5% (blue dashed line) and 1% (red dashed line) significance levels based on a Normal distribution for a two-sided test of equal forecast accuracy.

According to figure 8, the ensemble forecast $\{\hat{y}_{ens,t+24}\}_{t=1}^{P}$ appears to be at least as good as any of the forecasts $\{\hat{y}_{k,t+24}\}_{t=1}^{P}$, for $k = 1, ..., 1400$. In fact, at the 5% confidence level, the majority of the forecasts shows significantly worse performance than the ensemble forecast, while the remaining forecasts are as good as the ensemble. This finding is reported for all models except

[19]The spectral density estimator considered may in rare occasions take on negative values. In the present application, this is the case for approximately 0.3% of the K estimated forecasts. I follow Diebold and Mariano (1995) and treat the estimator as 0 for these cases which implies in an automatic rejection of the null hypothesis of equal forecast accuracy. Because in this application I am interested in a specific value for the statistic $\Delta_{k,h}$, I set it at $[4 \times sign(\bar{d}_{k,h})]$, which corresponds to a 4 standard deviation from the mean of a standard normal, scaled by the sign of the sample mean loss differential.

Figure 8: Distribution of the DM statistic Δ over different initializations

This figure plots the test statistic $\Delta_{k,24}$, for $k = 1, ..., 1400$, for each neural network model. The statistic $\Delta_{k,24}$ is defined such as positive values imply a better forecast performance of the ensemble prediction $\{\hat{y}_{ens,t+24}\}_{t=1}^{P}$ over the individual forecast $\{\hat{y}_{k,t+24}\}_{t=1}^{P}$, where different forecasts k are associated with different initializations. Forecasts are computed over the out-of-sample period of size P. The vertical dashed lines represent critical values at the 5% (blue) and 1% (red) significance levels based on a Normal distribution for a two-sided test of equal forecast accuracy.



the FF-pool, where the majority of forecasts does as good as the ensemble while a small amount of them shows superior performance. All in all, these results are not very surprising as they support the claim that ensemble estimators help improving forecast accuracy in the presence of models with intrinsic high variance. In the case of neural networks, the variance across predictions is related to the uncertainty over initial values, which is reduced substantially with the use of ensembles. Additionally, due to the highly nonlinear setting, it is plausible that the best initial values change over time. Combining the outcome of different forecasts in an ensemble-like approach is therefore a way of making the final prediction more robust to this type of uncertainty.

As pointed before, note that the ensemble prediction is less attractive for the FF-pool than it is for all the other models. This is likely related to its large amount of estimated parameters ($9\times$ compared to the FF-cpi and FF-LSTM, and $15\times$ compared to the LSTM-pool and LSTM-all; see

table 3 in appendix C). As such, the contribution of each single parameter tends to be quite small hence diminishing the initialization effect on the prediction. Also note that interestingly LSTM models appear to be less sensitive to initial values than feed-forward networks, with the exception of the FF-pool, as discussed.

4 Conclusions

This paper examines the suitability of neural networks to forecast inflation. To this end, it analyses the out-of-sample forecasting performance of a number of different neural network specifications with respect to standard benchmarks, and carries in-sample analysis as an attempt to clarify their behaviour. I consider the estimation of a feed-forward neural network as well as a recurrent neural network with long-short term memory (LSTM) units, the later being a novelty in the literature of inflation forecasting. The recurrent neural network is specially attractive for the task given its ability to combine dimensional reduction with long memory information under a highly nonlinear setting. In an empirical analysis with monthly US data, I distinguish between specifications containing data on inflation only, on a pool of economic predictors excluding CPI data, and a combination of both. This modelling choice is a way of isolating the effect of other economic predictors on the forecast.

The main results suggest that the LSTM model have advantages in predicting the long-term trend of inflation with respect to the more traditional feed-forward network (and standard benchmarks). This finding can be rationalized by the ability of the LSTM in incorporating information from long sequences of past observations in the prediction, while economizing in the number of estimated parameters. Additionally, the results point to an important role for macroeconomic information during periods of high economic uncertainty, in line with previous evidence. Throughout the period covering the Great Recession and its aftermath, the out-of-sample accuracy of neural networks is superior to standard linear benchmarks, pointing to a possible role for nonlinearities during this episode. Moreover, an in-sample analysis of the LSTM model demonstrates that the estimated common components of variation among the macro predictors capture well the dynamics of the business cycle. Finally, data on output, income and consumption are found to be important predictors of inflation.

References

Anders, U., and O. Korn. (1996). "Model selection in neural networks." In "CZEW Discussion Papers 96-21," .

Athey, S.. (2019). "The Impact of Machine Learning on Economics." In "Ajay Agrawal, Joshua Gans, and Avi Goldfarb (Eds.), The Economics of Artificial Intelligence: An Agenda," 507–547, Chicago: University of Chicago Press.

Atkeson, A., and L. E. Ohanian. (2001). "Are Phillips curves useful for forecasting inflation?" Quarterly Review, Federal Reserve Bank of Minneapolis 2–11.

Barnett, W., A. Medio, and A. Serletis. (2015). "Nonlinear and Complex Dynamics in Economics." Macroeconomic Dynamics 19, 8, 1749–1779.

Carriero, A., A.B. Galvão, and G. Kapetanios. (2019). "A comprehensive evaluation of macroeconomic forecasting methods." International Journal of Forecasting 35, 4, 1226–1239.

Chakraborty, C., and A. Joseph. (2017). "Machine learning at central banks." Bank of England Working Papers 674.

Cook, T. R., and S. Hall. (2017). "Macroeconomic Indicator Forecasting with Deep Neural Networks." In "Federal Reserve Bank of Kansas City, Research Working Paper 17-11," .

Coulombe, P. G., M. Leroux, D. Stevanovic, and S. Surprenant. (2020). "How is Machine Learning Useful for Macroeconomic Forecasting?" ArXiv 2008.12477.

Cybenko, G. (1989). "Approximation by superposition of a sigmoidal function." Mathematics of Control, Signals and Systems 2, 303–314.

Diebold, F., and R. Mariano. (1995). "Comparing Predictive Accuracy." Journal of Business & Economic Statistics 13, 3, 253–263.

Gaier, A., and D. Ha. (2019). "Weight Agnostic Neural Networks." `https://weightagnostic.github.io`.

Giacomini, R., and B. Rossi. (2010). "Forecast Comparisons in Unstable Environments." Journal of Applied Econometrics 25, 595–620.

Giannone, D., M. Lenza, and G. Primiceri. (2015). "Prior selection for vector autoregressions." Review of Economics and Statistics 97, 2, 436–451.

Giannone, D., M. Lenza, and G. Primiceri. (2018). "Economic predictions with big data: The illusion of sparcity." Working paper, Northwestern University .

Glorot, X., and Y. Bengio. (2010). "Understanding the difficulty of training deep feedforward neural networks." Journal of Machine Learning Research - Proceedings Track 9, 249–256.

Goodfellow, I. J., Y. Bengio, and A. Courville (2016). Deep Learning. MIT Press, `http://www.deeplearningbook.org`.

Gu, S., B. Kelly, and D. Xiu. (2019). "Empirical Asset Pricing via Machine Learning." In "Chicago Booth Research Paper No. 18-04; 31st Australasian Finance and Banking Conference 2018; Yale ICF Working Paper No. 2018-09.", .

Hanin, B., and D. Rolnick. (2018). "How to start training: The effect of initialization and architecture." In "Advances in Neural Information Processing Systems 31," 569–579, Curran Associates, Inc.

Hasenzagl, T., F. Pellegrino, L. Reichlin, and G. Ricco. (2018). "A Model of the Fed's View on Inflation." Science Po OFCE Working Paper , 3.

Hazell, J., J. Herreno, E. Nakamura, and J. Steinsson. (2020). "The Slope of the Phillips Curve: Evidence from U.S. States." NBER Working Paper No 28005 .

He, K., X. Zhang, Ren S., and J. Sun. (2015). "Delving deep into rectifiers: surpassing human-level performance on imagenet classification." In "Proceedings of the IEEE international conference on computer vision.", 1026–1034.

Ioffe, S., and C. Szegedy. (2015). "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift." ArXiv abs/1502.03167.

Jain, P., and P. Kar. (2017). "Non-convex Optimization for Machine Learning." Foundations and Trends in Machine Learning 10, 3-4, 142–336.

Kingma, D., and J. Ba. (2015). "Adam: A method for stochastic optimization." In "ICLR," .

Ludvigson, S., and S. Ng. (2007). "The empirical risk return relation: A factor analysis approach." Journal of Financial Economics 83, 1, 171–222.

McCracken, M. W., and S. Ng. (2016). "FRED-MD: A monthly database for Macroeconomic Research." Journal of Business & Economic Statistics 34, 4, 574–589.

Medeiros, M. C., G. Vasconcelos, A. Veiga, and E. Zilberman. (2019). "Forecasting Inflation in a Data-Rich Environment: The Benefits of Machine Learning Methods." Journal of Business & Economic Statistics 1–45.

Moody, J., and J. Utans. (1995). "Architecture selection strategies for neural networks: application to bond rating prediction." In "Refenes, A.-P.N. (ed.), Neural Networks in the Capital Markets," New York: Wiley.

Mullainathan, S., and J. Spiess. (2017). "Machine Learning: An Applied Econometric Approach." Journal of Economic Perspectives 31, 2, 87–106.

Nakamura, E.. (2005). "Inflation forecasting using a neural network." Economics Letters 86, 373–378.

Refenes, A. N., and A. D. Zapranis. (1999). "Neural Model Identification, Variable Selection and Model Adequacy." Journal of Forecasting 18, 299–332.

Refenes, A.P., and H. White. (1998). "Neural Networks and Financial Economics." International Journal of Forecasting 17, 347–495.

Schmidt-Hieber, J.. (2017). "Nonparametric regression using deep neural networks with ReLU activation function." .

Sermpinis, G., C. Stasinakis, K. Theofilatos, and A. Karathanasopoulos. (2014). "Inflation and Unemployment Forecasting with Genetic Support Vector Regression." Journal of Forecasting 33, 471–487.

Stock, J. H., and M. W. Watson. (1999). "Forecasting Inflation." Journal of Monetary Economics 44, 293–335.

Stock, J. H., and M. W. Watson. (2002). "Macroeconomic forecasting with diffusion indexes." Journal of Business & Economic Statistics 20, 147–162.

Stock, J. H., and M. W. Watson. (2007). "Why Has U.S. Inflation Become Harder to Forecast?" Journal of Money, Credit and Banking. 39, 7, 1849–1849.

Stock, J. H., and M. W. Watson. (2009). "Phillips Curve Inflation Forecasts." In "Fuhrer J, Kodrzycki Y, Little J, Olivei G Understanding Inflation and the Implications for Monetary Policy.", 99–202, Cambridge: MIT Press.

Stock, J. H, and M. W. Watson. (2019). "Slack and Cyclically Sensitive Inflation." NBER Working Paper No 25987 .

Stone, M.. (1974). "Cross-Validatory choice and assessement of statistical predictions." Journal of the Royal Statistical Society 36, 2, 111–147.

Varian, H. R.. (2014). "Big data: New tricks for econometrics." The Journal of Economic Perspectives 28, 2, 3–27.

A Data description

Table 2: Data description

The table describes the data used in the empirical analysis, collected from the FRED monthly database on November 2019. I follow McCracken and Ng (2016) and divide the series into eight economic groups. The column Tcode refers to the transformation applied to each series x_t, where (1) no transformation, (2) Δx_t, (3) $\Delta^2 x_t$, (4) $log(x_t)$, (5) $\Delta log(x_t)$, (6) $\Delta^2 log(x_t)$, (7) $\Delta(x_t/x_{t-1} - 1.0)$. The comparable series in Global Insight is given in the column GSI.

	Tcode	Fred mnemonics	Description	GSI	GSI: description
Group 1: Output and income					
1	5	RPI	Real Personal Income	M_14386177	PI
2	5	W875RX1	Real personal income ex transfer receipts	M_145256755	PI less transfers
3	5	INDPRO	IP Index	M_116460980	IP: total
4	5	IPFPNSS	IP: Final Products and Nonindustrial Supplies	M_116460981	IP: products
5	5	IPFINAL	IP: Final Products (Market Group)	M_116461268	IP: final prod
6	5	IPCONGD	IP: Consumer Goods	M_116460982	IP: cons gds
7	5	IPDCONGD	IP: Durable Consumer Goods	M_116460983	IP: cons dble
8	5	IPNCONGD	IP: Nondurable Consumer Goods	M_116460988	IP: cons nondble
9	5	IPBUSEQ	IP: Business Equipment	M_116460995	IP: bus eqpt
10	5	IPMAT	IP: Materials	M_116461002	IP: matls
11	5	IPDMAT	IP: Durable Materials	M_116461004	IP: dble matls
12	5	IPNMAT	IP: Nondurable Materials	M_116461008	IP: nondble matls
13	5	IPMANSICS	IP: Manufacturing (SIC)	M_116461013	IP: mfg
14	5	IPB51222s	IP: Residential Utilities	M_116461276	IP: res util
15	5	IPFUELS	IP: Fuels	M_116461275	IP: fuels
16	2	CUMFNS	Capacity Utilization: Manufacturing	M_116461602	Cap util
Group 2: Labor market					
17	2	HWI	Help-Wanted Index for United States		Help wanted indx

Continued on next page

Table 2 – *Continued*

	Tcode	Fred mnemonics	Description	GSI	GSI: description
18	2	HWIURATIO	Ratio of Help Wanted/No. Unemployed	M_110156531	Help wanted/unemp
19	5	CLF16OV	Civilian Labor Force	M_110156467	Emp CPS total
20	5	CE16OV	Civilian Employment	M_110156498	Emp CPS nonag
21	2	UNRATE	Civilian Unemployment Rate	M_110156541	U: all
22	2	UEMPMEAN	Average Duration of Unemployment (Weeks)	M_110156528	U: mean duration
23	5	UEMPLT5	Civilians Unemployed - Less Than 5 Weeks	M_110156527	U < 5 wks
24	5	UEMP5TO14	Civilians Unemployed for 41760 Weeks	M_110156523	U 41760 wks
25	5	UEMP15OV	Civilians Unemployed - 15 Weeks & Over	M_110156524	U 15+ wks
26	5	UEMP15T26	Civilians Unemployed for 15-26 Weeks	M_110156525	U 15-26 wks
27	5	UEMP27OV	Civilians Unemployed for 27 Weeks and Over	M_110156526	U 27+ wks
28	5	CLAIMSx	Initial Claims	M_15186204	UI claims
29	5	PAYEMS	All Employees: Total nonfarm	M_123109146	Emp: total
30	5	USGOOD	All Employees: Goods-Producing Industries	M_123109172	Emp: gds prod
31	5	CES1021000001	All Employees: Mining and Logging: Mining	M_123109244	Emp: mining
32	5	USCONS	All Employees: Construction	M_123109331	Emp: const
33	5	MANEMP	All Employees: Manufacturing	M_123109542	Emp: mfg
34	5	DMANEMP	All Employees: Durable goods	M_123109573	Emp: dble gds
35	5	NDMANEMP	All Employees: Nondurable goods	M_123110741	Emp: nondbles
36	5	SRVPRD	All Employees: Service-Providing Industries	M_123109193	Emp: services
37	5	USTPU	All Employees: Trade, Transportation & Utilities	M_123111543	Emp: TTU
38	5	USWTRADE	All Employees: Wholesale Trade	M_123111563	Emp: wholesale
39	5	USTRADE	All Employees: Retail Trade	M_123111867	Emp: retail
40	5	USFIRE	All Employees: Financial Activities	M_123112777	Emp: FIRE
41	5	USGOVT	All Employees: Government	M_123114411	Emp: Govt
42	1	CES0600000007	Avg Weekly Hours : Goods-Producing	M_140687274	Avg hrs
43	2	AWOTMAN	Avg Weekly Overtime Hours : Manufacturing	M_123109554	Overtime: mfg

Continued on next page

Table 2 – *Continued*

	Tcode	Fred mnemonics	Description	GSI	GSI: description
44	1	AWHMAN	Avg Weekly Hours : Manufacturing	M_14386098	Avg hrs: mfg
45	6	CES0600000008	Avg Hourly Earnings : Goods-Producing	M_123109182	AHE: goods
46	6	CES2000000008	Avg Hourly Earnings : Construction	M_123109341	AHE: const
47	6	CES3000000008	Avg Hourly Earnings : Manufacturing	M_123109552	AHE: mfg
Group 3: Housing					
48	4	HOUST	Housing Starts: Total New Privately Owned	M_110155536	Starts: nonfarm
49	4	HOUSTNE	Housing Starts, Northeast	M_110155538	Starts: NE
50	4	HOUSTMW	Housing Starts, Midwest	M_110155537	Starts: MW
51	4	HOUSTS	Housing Starts, South	M_110155543	Starts: South
52	4	HOUSTW	Housing Starts, West	M_110155544	Starts: West
53	4	PERMIT	New Private Housing Permits (SAAR)	M_110155532	BP: total
54	4	PERMITNE	New Private Housing Permits, Northeast (SAAR)	M_110155531	BP: NE
55	4	PERMITMW	New Private Housing Permits, Midwest (SAAR)	M_110155530	BP: MW
56	4	PERMITS	New Private Housing Permits, South (SAAR)	M_110155533	BP: South
57	4	PERMITW	New Private Housing Permits, West (SAAR)	M_110155534	BP: West
Group 4: Consumption, orders and inventories					
58	5	DPCERA3M086SBEA	Real personal consumption expenditures	M_123008274	Real Consumption
59	5	CMRMTSPLx	Real Manu. and Trade Industries Sales	M_110156998	M&T sales
60	5	RETAILx	Retail and Food Services Sales	M_130439509	Retail sales
61	5	ACOGNO	New Orders for Consumer Goods	M_14385863	Orders: cons gds
62	5	AMDMNOx	New Orders for Durable Goods	M_14386110	Orders: dble gds
63	5	ANDENOx	New Orders for Nondefense Capital Goods	M_178554409	Orders: cap gds
64	5	AMDMUOx	Unfilled Orders for Durable Goods	M_14385946	Unf orders: dble
65	5	BUSINVx	Total Business Inventories	M_15192014	M&T invent
66	2	ISRATIOx	Total Business: Inventories to Sales Ratio	M_15191529	M&T invent/sales
67	2	UMCSENTx	Consumer Sentiment Index	hhsntn	Consumer expect

Continued on next page

Table 2 – *Continued*

	Tcode	Fred mnemonics	Description	GSI	GSI: description
Group 5: Money and credit					
68	6	M1SL	M1 Money Stock	M_110154984	M1
69	6	M2SL	M2 Money Stock	M_110154985	M2
70	5	M2REAL	Real M2 Money Stock	M_110154985	M2 (real)
71	6	BOGMBASE	Monetary Base	M_110154995	MB
72	6	TOTRESNS	Total Reserves of Depository Institutions	M_110155011	Reserves tot
73	7	NONBORRES	Reserves Of Depository Institutions	M_110155009	Reserves nonbor
74	6	BUSLOANS	Commercial and Industrial Loans	BUSLOANS	C&I loan plus
75	6	REALLN	Real Estate Loans at All Commercial Banks	BUSLOANS	DC&I loans
76	6	NONREVSL	Total Nonrevolving Credit	M_110154564	Cons credit
77	2	CONSPI	Nonrevolving consumer credit to Personal Income	M_110154569	Inst cred/PI
78	6	MZMSL	MZM Money Stock	N.A.	N.A.
79	6	DTCOLNVHFNM	Consumer Motor Vehicle Loans Outstanding	N.A.	N.A.
80	6	DTCTHFNM	Total Consumer Loans and Leases Outstanding	N.A.	N.A.
81	6	INVEST	Securities in Bank Credit at All Commercial Banks	N.A.	N.A.
Group 6: Interest and exchange rates					
82	2	FEDFUNDS	Effective Federal Funds Rate	M_110155157	Fed Funds
83	2	CP3Mx	3-Month AA Financial Commercial Paper Rate	CPF3M	Comm paper
84	2	TB3MS	3-Month Treasury Bill:	M_110155165	3 T-bill
85	2	TB6MS	6-Month Treasury Bill:	M_110155166	6 T-bill
86	2	GS1	1-Year Treasury Rate	M_110155168	1 T-bond
87	2	GS5	5-Year Treasury Rate	M_110155174	5 T-bond
88	2	GS10	10-Year Treasury Rate	M_110155169	10 T-bond
89	2	AAA	Moody's Seasoned Aaa Corporate Bond Yield		Aaa bond
90	2	BAA	Moody's Seasoned Baa Corporate Bond Yield		Baa bond
91	1	COMPAPFFx	3-Month Commercial Paper Minus FEDFUNDS		CP-FF spread

Continued on next page

Table 2 – *Continued*

	Tcode	Fred mnemonics	Description	GSI	GSI: description
92	1	TB3SMFFM	3-Month Treasury C Minus FEDFUNDS		3 mo-FF spread
93	1	TB6SMFFM	6-Month Treasury C Minus FEDFUNDS		6 mo-FF spread
94	1	T1YFFM	1-Year Treasury C Minus FEDFUNDS		1 yr-FF spread
95	1	T5YFFM	5-Year Treasury C Minus FEDFUNDS		5 yr-FF spread
96	1	T10YFFM	10-Year Treasury C Minus FEDFUNDS		10 yr-FF spread
97	1	AAAFFM	Moody's Aaa Corporate Bond Minus FEDFUNDS		Aaa-FF spread
98	1	BAAFFM	Moody's Baa Corporate Bond Minus FEDFUNDS		Baa-FF spread
99	5	TWEXAFEGSMTHx	Trade Weighted U.S. Dollar Index		Ex rate: avg
100	5	EXSZUSx	Switzerland / U.S. Foreign Exchange Rate	M_110154768	Ex rate: Switz
101	5	EXJPUSx	Japan / U.S. Foreign Exchange Rate	M_110154755	Ex rate: Japan
102	5	EXUSUKx	U.S. / U.K. Foreign Exchange Rate	M_110154772	Ex rate: UK
103	5	EXCAUSx	Canada / U.S. Foreign Exchange Rate	M_110154744	EX rate: Canada
Group 7: Prices					
104	6	WPSFD49207	PPI: Finished Goods	M110157517	PPI: fin gds
105	6	WPSFD49502	PPI: Finished Consumer Goods	M110157508	PPI: cons gds
106	6	WPSID61	PPI: Intermediate Materials	M_110157527	PPI: int matls
107	6	WPSID62	PPI: Crude Materials	M_110157500	PPI: crude matls
108	6	OILPRICEx	Crude Oil, spliced WTI and Cushing	M_110157273	Spot market price
109	6	PPICMM	PPI: Metals and metal products:	M_110157335	PPI: nonferrous
110	5	CPIAUCSL	CPI : All Items	M_110157323	CPI-U: all
111	6	CPIAPPSL	CPI : Apparel	M_110157299	CPI-U: apparel
112	6	CPITRNSL	CPI : Transportation	M_110157302	CPI-U: transp
113	6	CPIMEDSL	CPI : Medical Care	M_110157304	CPI-U: medical
114	6	CUSR0000SAC	CPI : Commodities	M_110157314	CPI-U: comm.
115	6	CUSR0000SAD	CPI : Durables	M_110157315	CPI-U: dbles
116	6	CUSR0000SAS	CPI : Services	M_110157325	CPI-U: services

Continued on next page

Table 2 – *Continued*

	Tcode	Fred mnemonics	Description	GSI	GSI: description
117	6	CPIULFSL	CPI : All Items Less Food	M_110157328	CPI-U: ex food
118	6	CUSR0000SA0L2	CPI : All items less shelter	M_110157329	CPI-U: ex shelter
119	6	CUSR0000SA0L5	CPI : All items less medical care	M_110157330	CPI-U: ex med
120	6	PCEPI	Personal Cons. Expend.: Chain Index	gmdc	PCE defl
121	6	DDURRG3M086SBEA	Personal Cons. Exp: Durable goods	gmdcd	PCE defl: dlbes
122	6	DNDGRG3M086SBEA	Personal Cons. Exp: Nondurable goods	gmdcn	PCE defl: nondble
123	6	DSERRG3M086SBEA	Personal Cons. Exp: Services	gmdcs	PCE defl: service
Group 8: Stock market					
124	5	S&P 500	S&P's Common Stock Price Index: Composite	M_110155044	S&P 500
125	5	S&P: indust	S&P's Common Stock Price Index: Industrials	M_110155047	S&P: indust
126	2	S&P div yield	S&P's Composite Common Stock: Dividend Yield		S&P div yield
127	5	S&P PE ratio	S&P's Composite Common Stock: Price-Earnings Ratio		S&P PE ratio
128	1	VXOCLSx	VXO		

B On the estimation of neural networks

The parameters of neural network models are estimated by minimizing a loss function between the fitted and actual values over the in-sample period. In this application I use the mean squared error loss as specified in equation 2. Given that the function G is nonlinear with respect to the covariates x_t, the problem of minimizing the loss usually translates into optimizing a non-convex function. In these cases, iterative algorithms are more suitable than the classic optimization procedures applied to convex functions because of their properties that enforce the algorithm to rapidly converge to optima (Jain and Kar, 2017). The literature on neural networks usually applies gradient descent as an optimization method. Gradient descent is based on the property that, to minimize a given function $\mathcal{L}$, one needs to move in the direction of the negative gradient, -$\Delta_\Theta \mathcal{L}(\Theta)$. The parameters are then updated iteratively, such that $\Theta_i = \Theta_{i-1} - \alpha \Delta_\Theta \mathcal{L}(\Theta_{i-1})$, where α is the learning rate, determining the size of the step, and i is the iteration. In this study, I apply an extension of the gradient descent, called Adam optimization algorithm (Kingma and Ba, 2015), that features an adaptive learning rate.

The computation of the gradient may involve a single, random picked observation (stochastic gradient descent), a sub-group of observations (minibatch), or even all available observations (batch gradient descent). For the purpose of this analysis, the number of observations to be included, called batch size, is defined by grid search. Another important concept in machine learning is an epoch, defined as the number of passes of all observations through the algorithm, and not to be confounded with the number of iterations. At each time the gradient is computed, the algorithm updates the parameters, what defines an iteration. Hence, for the case of batch gradient descent the number of epochs coincides with the number of iterations. However for both stochastic gradient descent and minibatch methods, the number of iterations exceeds the number of epochs. The ultimate number of epochs is a hyperparameter selected by grid search.

The estimation process of neural networks, based on incremental updates of the parameters, means that the choice of the initial parameters is an important one. First, assigning equal weights to different nodes implies that they account for the same information and are therefore redundant. Random initialization is popular because it breaks the symmetry in the network. Second, one should avoid imposing too high or too low initial weight values in order to prevent the vanishing (or exploding) gradient problem, mentioned in section 2.2. Modern approaches to parameter initialization rely on the idea that the variance of the activations (output of nodes) should be

similar across layers (Hanin and Rolnick, 2018). This literature suggests that parameters should therefore be randomly drawn from some zero-centered distribution with a specified variance, while biases are usually initialized with zeros. Common approaches are the Xavier, Glorot and He initializations (Glorot and Bengio, 2010, He et al., 2015). This application considers the Glorot initialization, in which initial weights are drawn from a specific uniform distribution.[20]

The non-convexity usually encountered in neural networks tends to increase the sensitivity of the learning algorithm to initial values. This means that in practice the model delivers a slightly different prediction every time it is re-estimated, given the random initialization. A common solution adopted by the empirical literature is to repeat the estimation a (large) number of times and average out the predictions, which significantly reduces the variance of the ensemble prediction and consequently the uncertainty around the initial value. Section 3.6 provides further insights on this method in the context of the present empirical exercise.

The non-convexity also means that the estimated model is not guaranteed to be globally optimal. In fact, multiple local minima and flatten regions are likely to be present in many practical problems. An intuitive interpretation of these phenomena relies on the architecture of neural networks. For instance, the intrinsic symmetry of these models implies that if two nodes swap places, the final prediction would remain the same while the weight vector would be permuted, which translates into multiple optima. Another possibility relates to the mutual dependence of weights through the network. For example, if a zero weight is assigned to a particular node, all weights leading to that node can take any value, in which case the set of optimal solutions contains flat regions. In both cases described above, the model is not identified.

Given the non-identifiability typically present in neural networks, the recent literature converged to a model selection method that does not rely on any probabilistic assumptions and therefore is not affected by identification problems, called cross-validation (Stone, 1974, Moody and Utans, 1995, Anders and Korn, 1996, Refenes and Zapranis, 1999). Cross-validation is the most generally applicable strategy for model selection with neural networks and involves the estimation of the so called prediction risk, defined in the words of Moody and Utans (1995) as the expected performance of an estimator in predicting new observations. This makes this method quite appropriate for deep learning problems because these in general are interested in a good prediction accuracy on unseen data, and not necessarily on the statistical relevance of a particular

[20]Glorot initialization: $W \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_q+n_{q+1}}}, \frac{\sqrt{6}}{\sqrt{n_q+n_{q+1}}}\right]$, where n_q is the number of input units to layer q.

covariate, in which case classical inference would play a more important role.

In fact, non-identification in neural networks seems to not affect forecast performance, as implied by the recent work of Gaier and Ha (2019). The authors propose a parameter agnostic neural network and show that model specifications with strong inductive biases towards a specific task can perform relatively well *without* training. This is an important finding because it questions the relative importance of estimated parameters compared to model specification, and suggests that the ultimate parameter value is not crucial to achieve relative good performances.

C Model specification

The estimation of a neural network requires selecting a number of hyperparameters via cross-validation. For a feed-forward model, this implies in for example choosing the combination of number of nodes and number of layers more appropriate to our data. Table 3 indicates the compiled list of hyperparameters specific to the models considered, as well as the candidate and optimal values of each hyperparemeter.

The cross-validation process is split into two stages. Stage 1 focuses on hyperparameters specific to the model's architecture, while stage 2 selects the hyperparameters related to the optimization procedure. For example, the number of nodes in the network would be selected in stage 1, while the batch size in stage 2. This method shrinks significantly the computational time compared to the option of selecting all hyperparameters at once. Moreover, previous tests (not reported) indicated that the relative performance of the models are not very sensitive to changes in the number of epochs or batch size. The grid search around these hyperparameters is nonetheless performed for robustness purposes in stage 2. I henceforth refer to "specification" as a particular selection of a set of hyperparameters.

During stage 1, I follow a step-by-step procedure: the FF-cpi and FF-pool models are estimated first, followed by the LSTM-pool and LSTM-all models. As explained below, the cross-validation on the FF-LSTM model only occurs in stage 2. Both FF-cpi and FF-pool are estimated over 64 different specifications, where I let vary the number of lags, the number of nodes in the feed-forward layer(s) and the number of hidden layers in the network, as indicated in table 3.

Second, I use the optimal selection of number of nodes as estimated from the feed-forward models as *fixed* hyperparameters in the cross-validation of the LSTM models (recall that this model also includes a feed-forward section stacked to the LSTM unit). A cross-validation is then

performed over 32 different specifications for the LSTM models, including the number of lags, the number of hidden layers and the number of hidden states (referred in the main text as the internal memory). Finally, I set as fixed in the FF-LSTM model the optimal specifications selected from the previous steps. More specifically, I set the number of lags L in the feed-forward part of the model as equal to the optimal value from the FF-cpi, and the number of lags L to be included in the LSTM unit as equal to the optimal value from the LSTM-pool model, as well as the optimal values of nodes and hidden states. The reason behind these choices relies on the similarities between the underlying model structures. The strategy of fixing hyperparameters based on optimal values of nested models facilitates the comparison between models and significantly decreases the computational time. Finally, during stage 2, the hyperparameters related to the optimization process are allowed to vary for all models.

Each model specification is evaluated over a so called validation set. First consider splitting the full sample with T observations between an in-sample period, of size R, and an out-of-sample period, of size P, such that $T = R + P + h$. The in-sample period is further split into two consecutive sets, the training and validation samples. Each specification is then estimated over the training sample and evaluated over the validation sample.[21] The corresponding performance is used to differentiate between specifications. Figure 9 provides an illustrative setup of the method.

More specifically, the cross-validation exercise is implemented as follows: (i) split the data into consecutive samples: training, validation (three different lengths) and out-of-sample set, known as test sample in the machine learning jargon;[22] (ii) for each specification, estimate the model over the training sample and predict over the validation sample; (iii) repeat this process 140 times, and compute the average prediction, defined as the series that averages out the predictions of the 140 series at each point in time; (iv) measure the performance of the average prediction over each of the three sections of the validation sample; and (v) choose the specification with best average performance over the three sections of the validation sample. The out-of-sample performance is then obtained by evaluating the best specification on unseen data, in which case the final prediction is the average over 1400 different prediction series.[23]

[21] The forecast performance is measured as the root mean squared error.

[22] Choosing the length of each sample can be quite arbitrary and ultimately depends on each application. Here I split the data such that approximately 60% of the total sample is devoted to training only, 20% to validation and 20% to testing. The precise splits are shown in figure 9.

[23] The number of iterations for cross-validation is much smaller than the one used for out-of-sample performance as a way of minimizing the computational time given the high number of specifications to estimate. The choice is nonetheless somewhat arbitrary, and the number of repetitions were ultimately set such that it is a multiple of the number of available processors (28).

I use a modified version of the more traditional k-fold cross-validation to account for time series characteristics. By estimating and cross-validating the model on consecutive samples it is possible to avoid the look-ahead bias, since the performance is measured only on future data. At each time step, during cross-validation and out-of-sample performance, the estimation set expands by one observation and the prediction is compared with the actual value. In addition, I evaluate each specification over three nested sub-samples of the validation sample, the longest being the full validation set, and choose the model with best average performance over the splits. This is a simple way to add robustness to the analysis, since it minimizes the chances of sample-dependent results. For instance, with a single validation set, the chosen model is the one with best performance over those particular observations, but there is no guarantee it is going to continue outperforming the alternatives once we move towards the out-of-sample set. In practice, the choice of the number of splits is quite arbitrary. Here the splits are selected such that the minimum sub-sample size comprises approximately four years of observations.

Figure 9: Illustrative setup of the cross-validation and out-of-sample forecasting

Part 1 (top) depicts the cross-validation over the validation sample for the one-step ahead model. For each specification of hyperparameters, the model is trained over the training sample, and at each step over the validation sample one more data point is added for estimation. The performance over the validation sample is then given by the average performance over three sub-samples: 1993M05 - 1997M10, 1993M05 - 2002M03, 1993M05 - 2006M07. Part 2 (middle) depicts the same cross-validation procedure but for a three-steps ahead model. Note that the predictions cover all data points in the validation set. Finally, part 3 (bottom) illustrates the out-of-sample forecasting for a one-step ahead model. The resulting performance is the one recorded for each specification. The model is re-estimated every 48 months for both the cross-validation and out-of-sample performance.

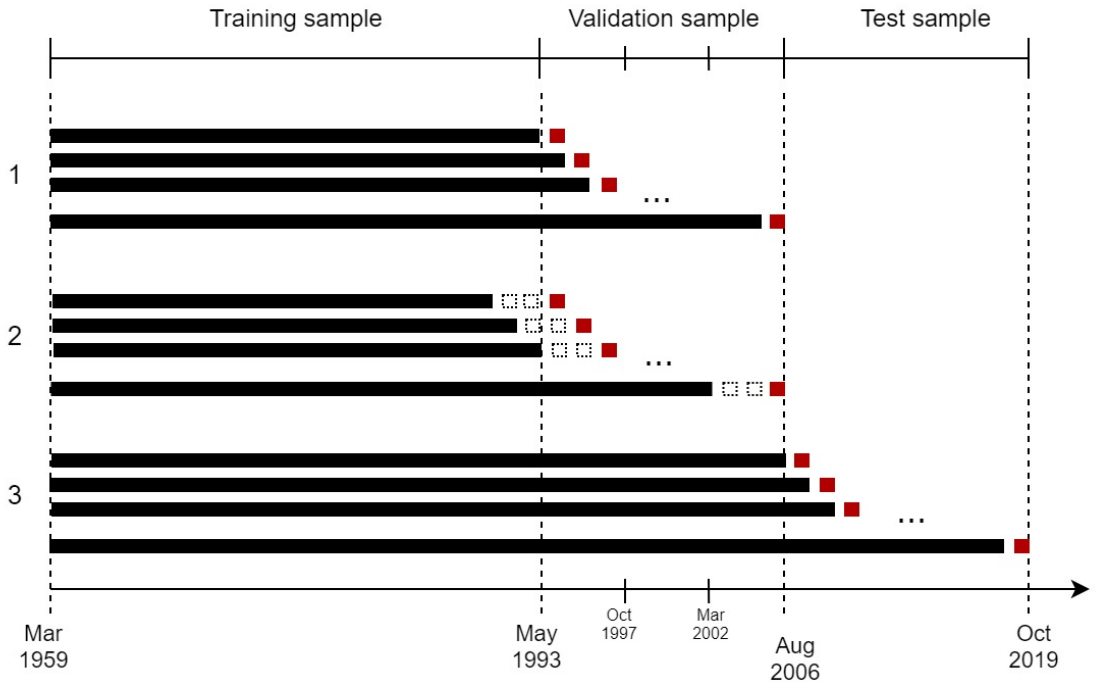


Table 3: Candidate and optimal values for hyperparameters

The table reports the candidate values for each hyperparameter as well as the optimal value selected by grid-search. The optimal hyperparameters of the FF-cpi and LSTM-pool selected in stage 1 are applied to the FF-LSTM model. Lag L imply that all the lags up to the specified number are included in the model. The batch size specified as *max* corresponds to the batch gradient descent method.

	FF-cpi		FF-pool		LSTM-pool		LSTM-all		FF-LSTM	
	Candidates	Optimal	Candidates	Optimal	Candidates	Optimal	Candidates	Optimal	Candidates	Optimal
Stage 1										
lags L	6,12,24,48	24	6,12,24,48	48	6,12,24,48	48	6,12,24,48	48		24, 48
nodes n	16, 32, 64, 128	128	16, 32, 64, 128	128	128	128	128	128		128
layers Q	1,2,3,4	4	1,2,3,4	3	3, 4	4	3, 4	4		4
$f_{t\|L}$-size p					2,4,6,8	2	2,4,6,8	2		2
# parameters		80513		758273		51017		51097		81737
Stage 2										
epochs	200,400,600	200	200,400,600	400	200,400,600	400	200,400,600	400	200,400,600	400
batch	128, *max*	128	128, *max*	128	128, *max*	*max*	128, *max*	*max*	128, *max*	128

D Benchmark specifications

Consider the inflation series $\pi_t = log(P_t) - log(P_{t-1})$, where P_t is the price index at time t (P_t corresponds to the series indexed by 110 in table 2).

D.1 Autoregressive model

I estimate an autoregressive (AR) model of order 1,

$$\pi_t = c + \Phi\pi_{t-1} + \nu_t, \qquad \nu_t \sim iid(0, \sigma^2)$$

for each horizon. The model is estimated by least squares and the h-step ahead forecast is

$$\widehat{\pi}_{t+h|t} = \frac{c\left(1 - \widehat{\Phi}^h\right)}{\left(1 - \widehat{\Phi}\right)} + \widehat{\Phi}^h \pi_t$$

D.2 Unobserved components with stochastic volatility (UC-SV)

A second benchmark is the UC-SV model from Stock and Watson (2007) which can be described as follows

$$\pi_t = \tau_t + e^{h_t/2}\varepsilon_t$$

$$\tau_t = \tau_{t-1} + u_t$$

$$h_t = h_{t-1} + v_t$$

where $\{\varepsilon_t\} \sim iid\mathcal{N}(0,1)$, $\{u_t\} \sim iid\mathcal{N}(0, \omega_\tau^2)$, and $\{v_t\} \sim iid\mathcal{N}(0, \omega_h^2)$. The state processes are initialized with $\tau_1 \sim \mathcal{N}(0, V_\tau)$ and $h_1 \sim \mathcal{N}(0, V_h)$, where $V_\tau = V_h = 0.12$. It is assumed independent inverse-gamma priors for ω_τ^2 and ω_h^2. The model is estimated using Markov Chain Monte Carlo (MCMC) methods, and the h-step ahead forecast is given by

$$\widehat{\pi}_{t+h|t} = \widehat{\tau}_t$$

D.3 Factor-augmented Distributed Lag (FADL) model

I specify a FADL(p) model for each horizon h, as in Carriero et al. (2019), such as

$$\pi_{t+h} = \beta_0 + \sum_{i=0}^{p-1} \beta_i \pi_{t-i} + \sum_{j=1}^{r} \gamma_j f_{j,t} + \nu_t$$

where r is the number of factors, and the number of factor lags is set to one. The factors are estimated by principal components applied to the data set of predictors $\mathbf{x_t}$ (see section 3.1 for more details on the data). The set $\mathbf{x_t}$ is previously transformed to guarantee stationarity following McCracken and Ng (2016), and then standardized. The point forecast is the average of the density forecast computed by fixed regressor bootstrap over 5000 replications.