Almosova, Anna; Andresen, Niek

Article — Published Version

Nonlinear inflation forecasting with recurrent neural networks

Journal of Forecasting

Provided in Cooperation with:
John Wiley & Sons

Suggested Citation: Almosova, Anna; Andresen, Niek (2022) : Nonlinear inflation forecasting with recurrent neural networks, Journal of Forecasting, ISSN 1099-131X, Wiley, Hoboken, NJ, Vol. 42, Iss. 2, pp. 240-259, https://doi.org/10.1002/for.2901

Received: 21 May 2021 | Revised: 14 January 2022 | Accepted: 25 April 2022

DOI: 10.1002/for.2901

RESEARCH ARTICLE

WILEY

Nonlinear inflation forecasting with recurrent neural networks

Anna Almosova[1] | **Niek Andresen**[2]

[1]Department of Economics and Law, Technische Universität Berlin, Berlin, Germany

[2]Department of Computer Engineering and Microelectronics, Technische Universität Berlin, Berlin, Germany

Correspondence
Anna Almosova, Technische Universität Berlin, Department of Economics and Law, Einsteinufer 17, 10587 Berlin, Germany.
Email: anna.almosova.hu.berlin@gmail.com

Funding information
Anna Almosova's work on this paper was partially supported by Schwerpunktprogramm 1764 of the German Science Foundation.

[Correction added on 7 September 2022, after first online publication: Projekt DEAL funding statement has been added.]

Abstract

Motivated by the recent literature that finds that artificial neural networks (NN) can efficiently predict economic time-series in general and inflation in particular, we investigate if the forecasting performance can be improved even further by using a particular kind of NN—a recurrent neural network. We use a long short-term memory recurrent neural network (LSTM) that was proven to be highly efficient for sequential data and computed univariate forecasts of monthly US CPI inflation. We show that even though LSTM slightly outperforms autoregressive model (AR), NN, and Markov-switching models, its performance is on par with the seasonal autoregressive model SARIMA. Additionally, we conduct a sensitivity analysis with respect to hyperparameters and provide a qualitative interpretation of what the networks learn by applying a novel layer-wise relevance propagation technique.

KEYWORDS

forecasting, inflation, LRP, LSTM, neural networks, SARIMA

1 | INTRODUCTION

Accurate inflation forecasting is essential for many economic decisions. Private investors predict future inflation to adjust their asset holdings, firms forecast the aggregate inflation level to adjust their prices and maximize profits, and central banks use inflation forecasts to conduct an efficient monetary policy. Fiscal authorities need to forecast inflation dynamics if they use the rate of inflation to adjust social security payments and income tax brackets.

Inflation forecasting is an interesting yet challenging task for both academic researchers and practitioners. As Stock and Watson (2007) point out, inflation in the United States has recently become both easier and harder to predict. On the one hand, since the mid-80s inflation became less volatile and as a result easier to predict. On the other hand, it became harder to outperform a naive univariate random walk-type forecast. For example, Atkeson and Ohanian (2001) show that averaging over the last 12 months gives a more accurate forecast of the 12-month-ahead inflation than a backward looking Phillips curve. Macroeconomic literature replies to this challenge by arguing that the inflation process might be changing over time. Consequently, a nonlinear model would give a more accurate inflation forecast.[1]

This paper evaluates the performance of a nonlinear nonparametric method from the machine learning

literature, that is novel in this context—a long short-term memory recurrent neural network (LSTM), which is a particular type of a neural network (NN). We see four main advantages of this method. First, LSTMs are flexible and data-driven. It means that the researcher does not have to specify the exact form of the nonlinearity. Instead the LSTM will infer it from the data itself. Second, as stated by the universal approximation theorem (Cybenko, 1989), under some mild regularity conditions LSTMs and neural networks of any type in general can approximate any continuous function arbitrarily accurately. At the same time, these models are more parsimonious than many other nonlinear time series models (Barron, 1993). Third, LSTMs were developed specifically for the sequential data analysis and were shown to be very successful with this task. In the machine learning literature, this method is widely applied in text analysis. For example, smartphones use LSTMs to predict the last word in the sentence and help the user while she is typing a message. Finally, the recent development of the optimization routines for NNs and the libraries that employ computer GPUs[2] made the training of NNs and recurrent neural networks (RNNs) significantly more feasible. Our main goal is to investigate if one can improve upon the accuracy of traditional forecasting models by using machine learning methods such as neural networks and, in particular, recurrent neural networks such as the LSTM. Out of all deep learning methods we decided to test the recurrent LSTM model because it is developed for the sequential data (time series) and because it can effectively find patterns even in long input sequences (when many data lags are used as input). Moreover, in contrast to classical time-series models, LSTM does not suffer from data instabilities and unit root problems. It can be applied to forecasting of any macroeconomic time-series in any country, provided that there is enough observations to estimate the model.[3]

Theoretically Convolutional Neural Networks—originally developed and successful in the domain of images—could also be used for time series forecasting if one treats the input as a one-dimensional image. We did not consider this method, because, similar to the presented simple NN, it does not take into account the input order explicitly like RNNs and LSTMs do. Very recently, new deep learning models for various forecasting scenarios have been introduced. DeepAR (Salinas et al., 2020) works well for situations with multiple time series with slightly different distributions, while here we only deal with inflation. The Spacetimetransformer (Grigsby et al., 2021) takes advantage of spatiotemporal data. In that case, the data points are not only related over time but also have neighbors in space like some measured values for cities. The Temporal Fusion Transformer model (Lim et al., 2021) also works with multiple time series for multiple horizon forecasting. These successful models have not been considered here as they deal with more complex data in different situations and are therefore more difficult to directly compare with the standard time-series models.

We compare the performance of an LSTM with a simple fully connected NN model as well as with several benchmark models that are frequently used for economic forecasting: a linear autoregressive model (AR), a random walk-type model (RW), a seasonal autoregressive model SARIMA and a Markov-switching autoregressive model (MS-AR). All our models are univariate (we leave multivariate forecasts for future research). We use monthly CPI inflation data for the US and compute indirect rolling forward monthly forecasts for up to 1 year ahead.

Neural networks are often criticized for their "black box" structure. Because NNs are nonlinear in parameters, it is difficult to interpret what they actually learn and how the input affects the output. We attempt to access the relevant importance of the model inputs for the final prediction by applying a layer-wise relevance propagation algorithm (LRP). The idea of this novel method is to decompose the final predicted value into the sum of the positive and negative values coming from the activated hidden units. The outcomes of the hidden units can in turn be decomposed into the contributions of the input neurons. This allows us to track how the value of a particular input contributes to the final prediction of the network.

One must note that the performance of neural networks is determined by several hyperparameters such as the number of hidden units or the learning rate. To provide some guidance on how to choose these parameters, we conduct an extensive sensitivity analysis. In total we train about 4300 different models.

According to our results, it is possible to systematically outperform the random walk forecasts. Mean squared forecast error (MSFE) of all tested models is approximately one third of the corresponding measure for the RW model at 1 to 12 months ahead. LSTM outperforms AR and NN models on the nonseasonally adjusted data but performs on par with the seasonal autoregressive model SARIMA. On the revised seasonally adjusted data all models—AR, SARIMA, MS-AR, NN, and LSTM—perform very similar. The exact ranking of the models varies depending on the deseasonalization procedure and the start of the forecast. We conclude from our analysis that LSTM is an efficient nonlinear model for forecasting inflation. Its usage, however, might not be justified in a univariate forecasting setting.

Our LRP analysis sheds some light on the differences between deseasonlized and nonseasonally adjusted data. For the nonseasonally adjusted data, LSTM seems to be able to learn the seasonal pattern on its own. For the adjusted data instead, LSTM learns a pattern that is very

similar to the pattern learned by the AR model. We also note that LSTM and NN models are better at predicting the dynamics after large shocks that bring inflation far away from its mean value. LSTM and NN correctly predict the magnitude of the mean reversion while SARIMA, AR and MS-AR models usually mistakenly expect inflation to stay away from its mean for a longer period.

One might wonder how difficult it is to set up a proper neural network model given that these models are very rich in hyperparameters.[4] It turns out that only a few of them meaningfully affect the performance of the neural networks on the inflation data. Based on our sensitivity analysis, we can conclude that the simple NN performs the best when Bayesian information criterion (BIC) is used for the lag length selection and when the maximum number of lags to select from is large. The performance of the NN is insensitive to the number of hidden units as long as there are more hidden units than the number of lags. The accuracy of the LSTM initially increases with the number of hidden units and then plateaus at around 100. For both models, the results are highly insensitive to the learning rate parameter. It is also important to train the LSTM for at least 500 epochs. In other words, when setting up an LSTM model for a forecasting exercise, the researcher should bear in mind two aspects: First, she should not include too few hidden units or allow for too few lags; second, she should train the model for a sufficient amount of time. Other hyperparameters appear not to be much of a concern.

This paper is not alone in applying deep learning methods to macroeconomic forecasting. Ahmed et al. (2010) and Stock and Watson (1998) compared linear and nonlinear methods for macroeconomic forecasting by averaging their performance over a large number of macro time series. Similar to our results, these studies find that simple NNs perform well at short forecasting horizons. However, they use a different optimization algorithm to train the NNs and a different procedure to select the test set. Other examples include Kuan and Liu (1995), who demonstrated the success of simple and recurrent NNs for the exchange rate forecasting in several countries; Swanson and White (1997b) and Swanson and White (1997a), who evaluated the advantage of the NNs with time varying coefficients in a real-time forecasting setup; and Kock and Teräsvirta (2011), who discuss direct versus indirect forecasts with NNs.

Chen et al. (2001), Nakamura (2005), and McAdam and McNelis (2005) discuss the inflation forecasting with neural networks. These studies show that NNs outperform benchmark linear models for shorter horizons based on various performance measures.

Our paper is different from the above mentioned literature in that, to the best of our knowledge, this is the first work that applies an LSTM recurrent neural network to inflation forecasting. Makridakis et al. (2018) do look at LSTM in their large study of the machine learning methods. They, however, consider an average performance across a large number of the data series and do not look on inflation in isolation. Their study indicates that machine learning methods do not outperform standard statistical methods on average. One of the most straightforward reasons for why their conclusion is in contrast with our findings might be the fact that nonlinear methods are not beneficial for all macro time-series. They, however, might be superior when one wants to forecasts inflation. For example, Stock and Watson (1999) found that at 12 months ahead horizon nonlinear methods perform the best for some macrovariables, including consumer prices, but not all of them. Another work that is closely related to our study is Elger et al. (2006) who consider a recurrent neural network but of a different class than the LSTMs analyzed here. They show that for shorter horizons, recurrent NNs are comparable with Markov switching autoregressive models and at longer horizons Markov switching models are more accurate. However, they apply a different type of RNN, different cross-validation and forecast evaluation procedures. Moreover, their study uses GDP inflation data while we focus on CPI inflation forecasting.

We also use a different optimization algorithm to fit the NN and RNN models than the existing literature—the Adam optimizer. Our choice of the optimization routine is based on its success in machine learning applications. While many of the existing papers decide on a particular architecture of the NN a priori, this study, on the contrary, carefully investigates how our conclusions about the comparison of different methods are affected by the hyperparameters of the NN and the LSTM. Finally, our LRP computation is novel to the macro forecasting literature. The discussion of LRP in the machine learning context can be found in Lapuschkin et al. (2016) and Arras et al. (2017). In a broader sense, this paper contributes to the literature on the nonlinear time-series forecasting. Teräsvirta (2006) and Teräsvirta and Case (2017) provide an overview of these methods.

We consider our work to be different from the usual "horse-race" exercise. Our goal is rather to investigate the novel machine learning method in more details. We comment on the specification selection and try to understand the mechanism behind the successful performance of the LSTM models. In this regard, our work is in line with the recent paper of Medeiros et al. (2018) that argues that machine learning methods offer a very promising toolkit for inflation predictions. They focus on multivariate random forest models. Conclusions similar to ours are also found in Smalter Hall and Cook (2017), who show that different types of NN outperform an autoregressive

model and perform better than the survey of professional forecasters for unemployment predictions.

The rest of the paper is organized as follows. Section 2 describes our set up and the forecasting models. It also gives a brief data description. Results are discussed in Section 3. Section 4.2 lays out the sensitivity analysis, and Section 4.3 presents the layer-wise relevance propagation analysis. Section 5 concludes.

2 | METHODOLOGY

We rank the forecasting methods based on the mean squared forecast error (MSFE) for out-of-sample forecasts on the test set. The forecast period starts in January 1990 and the observations from January 1960 until January 1990 are included into the training and validation set. Test set includes data from January 1990 until June 2020 and is never used for training or parameter tuning.

For the sensitivity analysis, we allocate a part of the training set as a validation set. The break point between training and validation set was chosen such that the number of observations in the validation set N^{val} is equal to 10% of the total number of observations in the whole training set.

We consider indirect (iterrolling forward) h-step-ahead forecasts. While in theory direct forecasts are more robust to model misspecifications, they are less efficient if the model is correctly specified. Marcellino et al. (2006) showed that in the linear forecasting setup, indirect forecasts typically perform better than direct ones. Kock and Teräsvirta (2011) address the same issue for nonlinear prediction methods. They conclude that iterated and direct forecast often have similar performance and their exact ranking is problem- and data-specific. Direct forecasts also require a separate model for each forecasting horizon and are thus more complex computationally. We focus on the indirect forecasts in this study and leave the extension to direct forecasts for our future research.

We are interested in the conditional forecast, which is made at date $t-1$. We start from the following model:

$$y_t = f(Z_{t,p}; W) + u_t, \text{with} Z_{t,p} = [y_{t-1}, ...y_{t-p}] \quad (1)$$

where y_t is a variable of interest, $f(\cdot)$ is an unknown, potentially nonlinear function, W is the matrix of model parameters (weights), Z collects the lags of y, and the number of lags p is chosen by an information criterion.

One and two step-ahead forecasts based on the information set F_{t-1} can be computed as

$$\hat{y}_{t|t-1} = \mathbb{E}[y_t|F_{t-1}] = f(Z_{t,p}; W) \quad (2)$$

$$\hat{y}_{t+1|t-1} = \mathbb{E}[y_{t+1}|F_{t-1}] = \mathbb{E}\left[f\left(\hat{y}_{t|t-1}, Z_{t,p-1}; W\right)|F_{t-1}\right] = \\ = \mathbb{E}[f(f(Z_{t,p}; W) + u_t, Z_{t,p-1}; W)|F_{t-1}] \quad (3)$$

Because the function $f(\cdot)$ is nonlinear, we cannot take the error term u_t out of the expectation operator as in the linear case. Ideally, one would need to estimate the expectation term by numerical integration. However, moving beyond the two-step-ahead forecast would require evaluating multiple integrals. Moreover, the assumption about the distribution of the u_t could matter for the result, and it would be hard to justify what this distribution should be. Finally, numerical integration or integration by bootstrap could introduce additional inefficiency.

We overcome this problem by assuming that the error term is zero in all states $u_t = 0$. It implies that our forecast is not an unbiased conditional mean estimator.[5] In other words, our nonlinear models receive solely the information about the first moment of the one-step-ahead forecast when they compute the $h \geq 2$ forecasts. It means that if anything, we only harm the performance of the nonlinear estimators. Our results can be seen as the lower bound of potential forecasting performance of the nonlinear methods.

2.1 | Overall procedure

The overall methodology of the paper is presented in the diagram below (Figure 1). Whenever the lag length needs to be selected for a particular model on a particular dataset, it is done in an AR model build with all the same parameters and on the same dataset.

2.2 | Forecasting models

1. RW: random walk-type forecasts are constructed as a simple average over the n previous periods (Atkeson & Ohanian, 2001; Stock & Watson, 2007).

$$\hat{y}_{t|t-1} = \frac{1}{n}\sum_{i=1}^{n} y_{t-i} \quad (4)$$

The results are presented for the standard random walk model that is $n = 1$. We also tried all $n \leq 12$ and obtain very similar results in terms of model comparison.

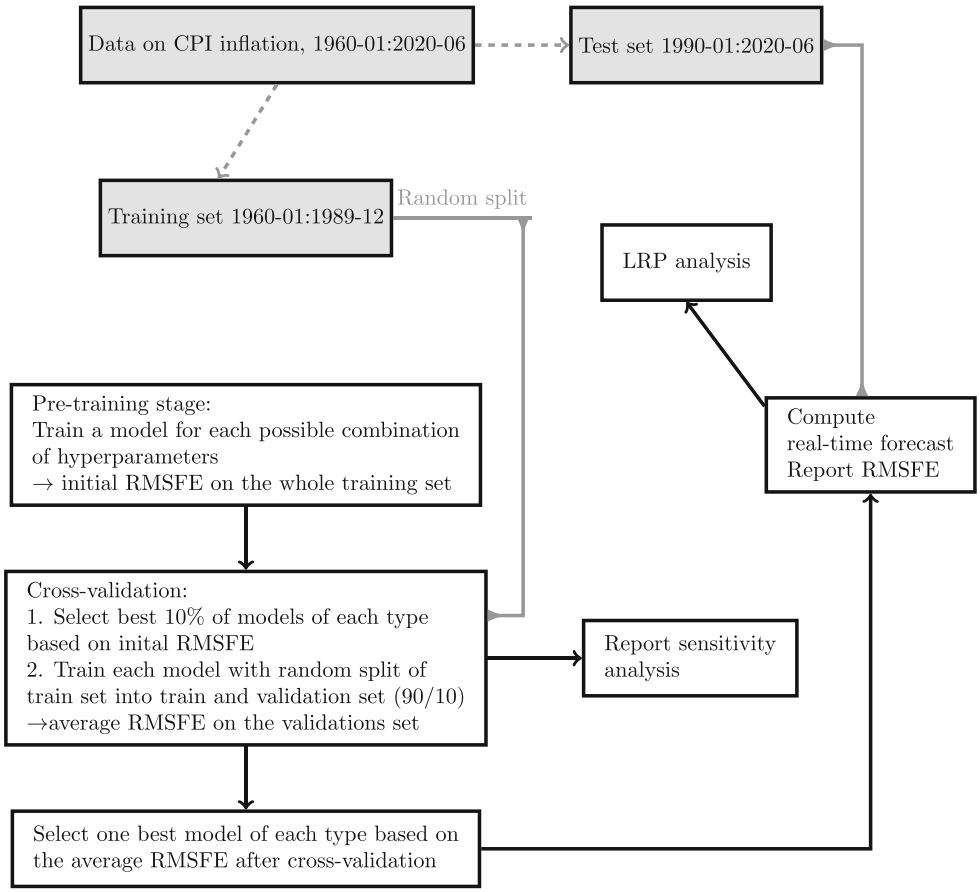

FIGURE 1 Flow diagram of the methodology

2. AR(p): univariate autoregession model of an order p.

$$\hat{y}_{t|t-1} = A + BZ_{t,p}, \text{with } Z_{t,p} = [y_{t-1}, \ldots y_{t-p}] \tag{5}$$

3. NN: simple fully connected neural network (Swanson & White, 1997b) with one hidden layer:

$$\hat{y}_{t|t-1} = b + \sum_{n=1}^{N} w_n \cdot \sigma\left(b^n + \sum_{\tau=1}^{P} w_\tau^n y_{t-\tau}\right) \tag{6}$$

where $\sigma(\cdot)$ is a nonlinear activation function,[6] b is a bias of the output and b^n is a bias of the hidden units n, w_n is a weight from the hidden unit n to the output, w_τ^n is a weight from the lag τ to the hidden unit n. Figure 2 sketches the structure of a simple NN model (biases are ignored).

4. LSTM: Long short-term memory recurrent neural network.

LSTM represents a particular type of a recurrent neural network (RNN), which is in turn a specific type of neural network (Figure 3). The difference between the NN and the recurrent neural network comes from the recurrent nature of the latter. An RNN represents an

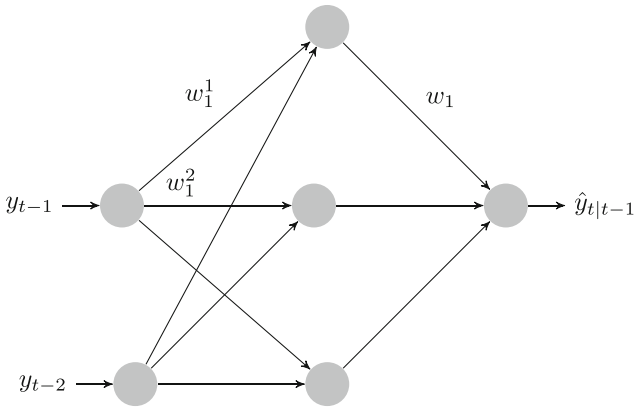

FIGURE 2 Fully connected neural network ($p = 2$) with no biases

NN model that receives one data lag as an input at a time and not all the lags at once as a vector. The prediction is computed and updated sequentially after seeing each of the data lags. Intermediate output, which is called the "state" of the network, is used as an additional input at the next time step. A representation of a standard recurrent neural network is given in Figure 4. Recurrent propagation of the state is

represented by the horizontal arrows. Note that the weights of the network stay the same, that is, the "RNN" block on the figure is identical for every time step. See also Smalter Hall and Cook (2017).

The RNN's structure has two important implications. First, the network is explicitly informed that the input lag y_{t-2} comes before the lag y_{t-1}. More recent lags are likely to be more important for the final prediction. This stands in contrast to the simple fully connected NN that treats all the lags equally such that the sequence of lags does not matter. Second, the network remembers information about the distant input lags when computing the final output. In text analysis, for example, if an RNN is used to predict the last word in the sentence, the first few words can inform the network about whether the sentiment of the sentence is positive or negative. In our application, the state of the RNN can potentially infer the information about trend, cycle, or seasonality. Another appealing feature of the RNN, which is, however, less relevant in our application, is that the input sequences $\{y_{t-\tau}\}_{\tau=1}^{p}$ do not have to be all of the same length, that is, p can be variable.

The LSTM network (Figure 5) is used in a sequence similar to the aforementioned RNN of which it is a special kind. It is distinct through its internal structure consisting of so called "gates." These allow the network to decide on its own what part of the network state and the input it wants to remember on the next iteration and what part it can forget. Such architecture leads to a better empirical performance (Hochreiter & Schmidhuber, 1997). A representation of an LSTM cell is given in Figure 6. This cell is used recursively as many times as there are data lags in the model.

Here, y_{t-1} represents the input at the time step t (e.g., a lagged inflation value). c is the "state" of the

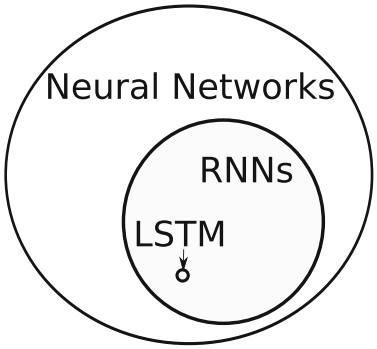

FIGURE 3 Classification of artificial neural networks

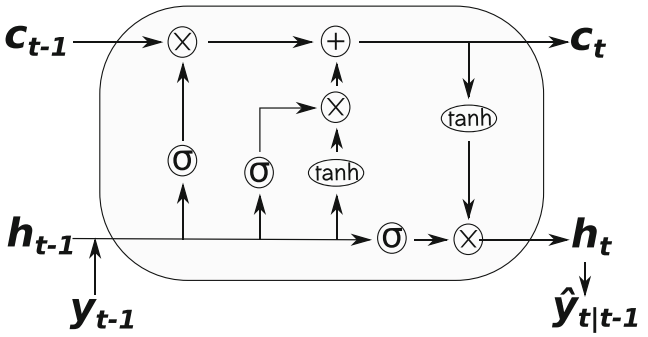

FIGURE 6 LSTM cell representation

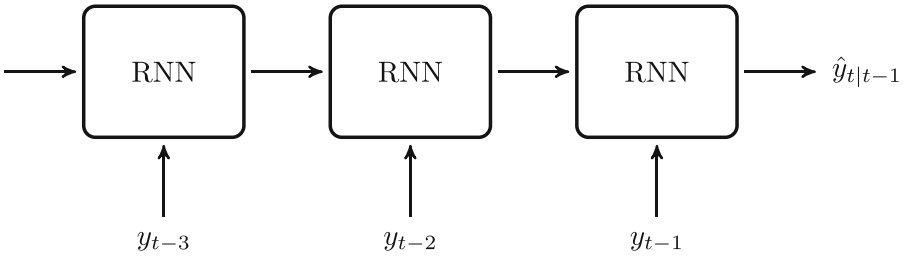

FIGURE 4 Basic representation of a recurrent neural network ($p = 3$)

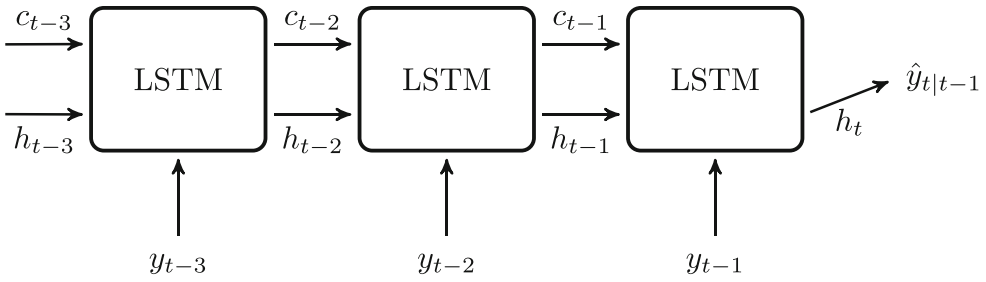

FIGURE 5 Basic representation of an LSTM recurrent neural network ($p = 3$)

network that represents its memory about the past. h_t denotes the output of the LSTM at the step t. $\sigma(\cdot)$ and $tanh(\cdot)$ represent the gates that are small neural networks that have the sigmoid function or the hyperbolic tangent function as activations at the output. On the left of the diagram, one can see that the prediction of the network for the period $t-1$ (h_{t-1}) and the input value at this step (y_{t-1}) are combined and then filtered through the four gates. These determine what part of the state c_{t-1} should be forgotten, what and how much information should be added to the state and how the prediction for the step t should be adjusted based on the state. On the right side of the diagram, the output of the cell is presented: It is a new updated value h_t, and $\hat{y}$ that is computed from it. For a detailed description of the LSTM architecture, we refer the reader to Appendix A as well as Hochreiter and Schmidhuber (1997).

5. Other nonlinear models: SARIMA $(p,d,q)(P,D,Q)_S$ and MS-AR.
   We additionally looked at two nonlinear time-series methods: A seasonal autoregressive model (SARIMA) and a Markov switching autoregressive model (MS-AR). The choice of the SARIMA model (Lütkepohl, 2005) is motivated by the fact, that the performance of the nonlinear neural network models might be resulting from their ability to capture seasonality and it is interesting to compare their performance with a linear seasonal model. Based on the ACF and PACF analysis, we select the SARIMA(1,1,1)(0,0,1)[12] model. We set the number of seasons to be 12 because we use monthly data.

The choice of the Markov switching model is motivated by the findings of Elger et al. (2006) that the Markov switching model is superior to the neural network in inflation forecasting at longer horizons. We use the same model specification, which is a two-state discrete Markov chain with regime-switching volatility and constant across regimes autoregressive parameters.

Both methods are trained with the maximum likelihood method implemented in the *statsmodels* library in Python.[7] Note that the SARIMA and MS-AR models are trained with a different training procedure than the rest of the models. Consequently, the differences in the forecasting performance of these two models result from both the optimization routine and the model structure. We would also like to stress that even though these models are more common than the neural networks, they are neither simple nor easy to fit. A number of issues such as model specification, selection of the hyperparameters and local optima are applicable to SARIMA and MS-AR.

2.3 | Optimization algorithm

All models (except the RW, SARIMA, and MS-AR) are trained by backpropagation with an adaptive stochastic gradient descent optimizer—Adam—whose success was largely documented in the machine learning literature (Kingma & Ba, 2014). Adam is different form the standard stochastic gradient descent algorithm in that it updates each parameter θ separately and changes the speed of the adjustment η depending on the "momentum" m_t or approximated first-order moment and the "friction" v_t or approximated second-order moment of the gradient g_t.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\cdot g_t \tag{7}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)\cdot g_t^2 \tag{8}$$

$$\hat{m}_t = \frac{m_t}{1-{\beta_1}^t} \tag{9}$$

$$\hat{v}_t = \frac{v_t}{1-{\beta_2}^t} \tag{10}$$

$$\theta_{t+1} = \theta_t - \eta\cdot\frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon} \tag{11}$$

Here β_1 and β_2 are tuning parameters of the estimator. Typical values are: $\beta_1 = 0.9, \quad \beta_2 = 0.999$.

An early stopping rule is not employed at training time. The grid search over the hyper-parameter space includes the number of training epochs as one of the dimensions. Thus, the choice of the best model there prevents overfitting in a similar way as an early stopping rule would. Because the standard frequentist procedure of fitting AR models does not include any regularization, we rewrite our AR in a simple NN form and train it in the same way as NN and LSTM models.[8] More specifically, we represent the AR model as a special case of a simple NN with a linear activation function and no hidden layer.

2.4 | Data

We use data on monthly US CPI inflation from the FRED database of the Federal Reserve Bank of St. Louis for 1960:01–2020:06 which constitutes 726 observations. Inflation rates are calculated as percentage change. The data is seasonally adjusted which is standard for the inflation forecasting literature. However, seasonally adjusted series are computed by the Bureau of Labor Statistics

with the help of seasonality filters that can favor nonlinear methods. More specifically, the seasonal adjustment is a two-sided nonlinear data transformation. Nonlinear methods can, potentially, achieve a good fit by simply unraveling the seasonal transformation. Additionally, the seasonally adjusted data is subject to annual revisions and thus cannot be used for our real-time forecasting exercise. We therefore, repeat our analysis for nonseasonally adjusted monthly and annual inflation rates as well as for the data with naive seasonal-adjustment (subtraction of historic monthly averages). Nonseasonally adjusted data allows testing the hypothesis that neural networks can learn the seasonality of the data on their own. Stock and Watson (1998) and Teräsvirta et al. (2005) are examples of forecasting exercise on a nonseasonally adjusted inflation data.

3 | RESULTS

We conduct a counterfactual exercise to answer the question "What would have happened if forecasters were using LSTM models instead of AR to predict inflation since the year 1990?" This question is particularly relevant from the policymakers' perspective since she has to make decisions in real-time and therefore needs forecasts based on the data available in real-**time.**

We iteratively compute inflation forecasts month by month. For example, estimations for the year 2001 are based solely on the data up until 2000. Forecasts for $h \geq 2$ are computed by iterating forward the one-step-ahead forecast.

Our findings for the real-time forecast for the nonseasonally adjusted data are presented in the Table 1 and can be summarized as follows.[9] First, all models outperform the naive random walk model. This holds true for using any number of lags between 1 and 12 for the computation of the RW forecast. Second, LSTM improves upon the performance of the regularized AR and of the NN models. It is however, on par with the SARIMA model for all horizons except for 1 and for 10–12 steps ahead where it is outperformed by SARIMA.

Table 2 contains the results for the seasonally adjusted data. On the deseasonalized data (upper panel of the table), forecast errors become smaller in absolute terms for all methods. In general, the performance of all methods becomes very close to each other. SARIMA uniformly outperforms other methods, although the improvement is not always statistically significant. This result is also not robust to the deseasonalizing procedure, data preprocessing and the start of the forecasting period. If we remove the season with monthly dummies (lower panel of the table), work with data in first differences or start the forecast at a date different from 01-1990, the differences in the performance of AR, LSTM, and SARIMA become negligible. Depending on the forecast horizon AR, LSTM, or SARIMA show the best performance. It is worth to keep in mind that seasonally adjusted CPI data is revised and thus the analysis do not represent a real-time forecasting exercise.

TABLE 1 Real-time forecasts errors, NA monthly inflation data

	MSFE						MAFE				
h	**RW**	**AR**	**NN**	**LSTM**	**MS**	**S**	**AR**	**NN**	**LSTM**	**MS**	**S**
1	0.321	0.119	0.118	0.107	0.102	**0.089**	0.255	0.255	0.236	0.239	**0.221**
2	0.374	0.117	0.121	0.115	0.138	**0.113**	0.249	0.253	**0.238**	0.274	0.244
3	0.396	0.110	0.111	**0.107**	0.144	0.115	0.237	0.235	**0.227**	0.28	0.243
4	0.397	0.110	0.107	**0.105**	0.141	0.112	0.235	0.23	**0.224**	0.275	0.237
5	0.403	0.110	0.11	**0.104**	0.137	0.110	0.235	0.235	**0.225**	0.265	0.232
6	0.402	0.110	0.113	**0.104**	0.134	0.109	0.237	0.239	**0.228**	0.263	0.232
7	0.399	0.111	0.112	**0.105**	0.131	0.108	0.237	0.237	**0.231**	0.261	**0.231**
8	0.398	0.109	0.112	**0.103**	0.131	0.107	0.236	0.239	**0.229**	0.263	0.234
9	0.384	0.107	0.109	**0.103**	0.126	0.104	0.235	0.233	**0.229**	0.262	0.23
10	0.363	0.105	0.106	0.102	0.12	**0.099**	0.231	0.231	0.227	0.255	**0.226**
11	0.338	0.108	0.108	0.106	0.119	**0.099**	0.234	0.234	0.231	0.252	**0.224**
12	0.349	0.115	0.114	0.111	0.122	**0.104**	0.246	0.244	0.239	0.256	**0.229**

Note: Mean squared forecast error (MSFE) and mean absolute forecast error (MAFE) are computed for real-time forecasts starting from January 1990 for nonseasonally adjusted monthly CPI inflation. The first column shows the forecast horizon, h. MS stands for Markov-switching model, S for seasonal SARIMA model, NN for neural network, LSTM for long short-term memory recurrent neural network. The lowest error in each row is highlighted in bold. MAFE for random walk model is not presented.

TABLE 2 Real-time forecasts errors, SA monthly inflation data

	MSFE					MAFE				
h	**AR**	**NN**	**LSTM**	**MS**	**S**	**AR**	**NN**	**LSTM**	**MS**	**S**
1	0.076	0.081	0.075	0.064	**0.06**	0.183	0.192	0.183	0.172	**0.163**
2	0.077	0.076	0.105	0.085	**0.075**	0.181	0.187	0.200	0.195	**0.179**
3	0.08	0.082	0.103	0.087	**0.075**	0.184	0.192	0.202	0.197	**0.177**
4	0.081	0.081	0.086	0.085	**0.073**	0.186	0.190	0.196	0.192	**0.171**
5	0.08	0.082	0.081	0.084	**0.074**	0.186	0.192	0.188	0.191	**0.174**
6	0.079	0.083	0.08	0.084	**0.074**	0.188	0.196	0.191	0.192	**0.173**
7	0.081	0.081	0.089	0.082	**0.072**	0.188	0.196	0.198	0.194	**0.177**
8	0.081	0.081	0.09	0.081	**0.072**	0.189	0.193	0.199	0.195	**0.179**
9	0.08	0.081	0.081	0.08	**0.07**	0.190	0.195	0.195	0.192	**0.175**
10	0.08	0.083	0.078	0.079	**0.068**	0.190	0.197	0.19	0.191	**0.172**
11	0.085	0.087	0.082	0.081	**0.07**	0.194	0.199	0.198	0.193	**0.174**
12	0.082	0.08	0.089	0.085	**0.074**	0.189	0.191	0.203	0.198	**0.179**
1	0.099	0.099	0.096	0.084	**0.079**	0.226	0.226	0.221	0.215	**0.205**
2	0.1	0.102	0.106	0.106	**0.098**	0.225	0.227	0.229	0.235	**0.221**
3	**0.095**	0.1	0.107	0.113	0.1	**0.214**	0.222	0.227	0.245	0.223
4	**0.095**	0.099	0.107	0.111	0.098	**0.213**	0.220	0.225	0.24	0.217
5	**0.095**	0.099	0.104	0.108	0.097	0.214	0.220	0.224	0.234	**0.213**
6	**0.095**	0.099	0.101	0.107	0.096	0.216	0.222	0.224	0.236	**0.213**
7	0.095	0.098	0.1	0.106	**0.094**	0.216	0.220	0.223	0.238	**0.212**
8	**0.093**	**0.094**	**0.094**	0.107	**0.093**	0.215	0.217	0.219	0.239	**0.213**
9	**0.091**	**0.09**	0.093	0.103	**0.09**	0.213	0.211	0.217	0.234	**0.209**
10	0.09	0.09	0.097	0.1	**0.086**	0.211	0.212	0.219	0.23	**0.206**
11	0.094	0.093	0.095	0.1	**0.087**	0.214	0.213	0.219	0.23	**0.207**
12	0.096	0.096	0.095	0.102	**0.091**	0.220	0.219	0.219	0.233	**0.211**

Note: Mean squared forecast error (MSFE) and mean absolute forecast error (MAFE) are computed for real-time forecasts starting from January 1990 for seasonally adjusted monthly CPI inflation. The upper panel corresponds to the standard seasonal filter, the lower panel corresponds to removing the season with monthly dummies. The first column shows the forecast horizon, h. MS stands for Markov-switching model, S for seasonal SARIMA model, NN for neural network, LSTM for long short-term memory recurrent neural network. The lowest error in each row is highlighted in bold.

The reason why all methods perform very similar on the deseasonalized data, while LSTM and SARIMA being more accurate on row data, might lie in the fact that SARIMA and LSTM learn the season on their own. In a univariate forecasting framework the ability to learn a season undoubtedly affects the model performance. We investigate this question further in the next section when we interpret what the LSTM networks have learned.

Figures 7–10 provide more insight into our results for seasonally adjusted data. Figure 7 depicts the errors of different models computed as MSFE. We see that all models behave similar and all of them have difficulties in predicting rare events such as the rapid spike in inflation after the Hurricane Katrina in 2005, the large decline in inflation in the second part of 2008 or in the first months of 2020.

As can be seen from the example on Figure 8, NN and LSTM tend to produce a forecast path (predictions for 1 to 12 months ahead) which is more nonlinear and more volatile than SARIMA and Markov-switching models. This nonlinearity can improve the forecasting performance but can also result in an extremely inaccurate "wild" forecast, as shown on Figure 9. Another observation is that NN, LSTM, and sometimes AR more accurately predict the magnitude of the mean reversion when starting the forecast from an extremely low or an extremely high level of inflation. Figure 10 demonstrates this point. MS-AR and SARIMA tend to underestimate the rise or decline in the first months and mistakenly

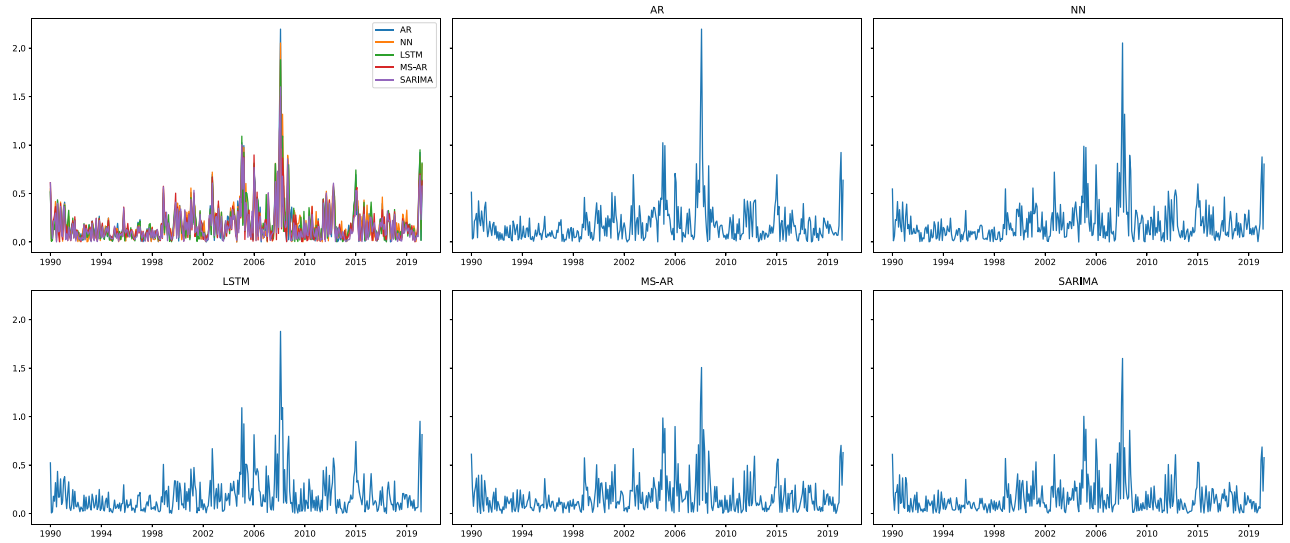

FIGURE 7 MSFE over time for different models

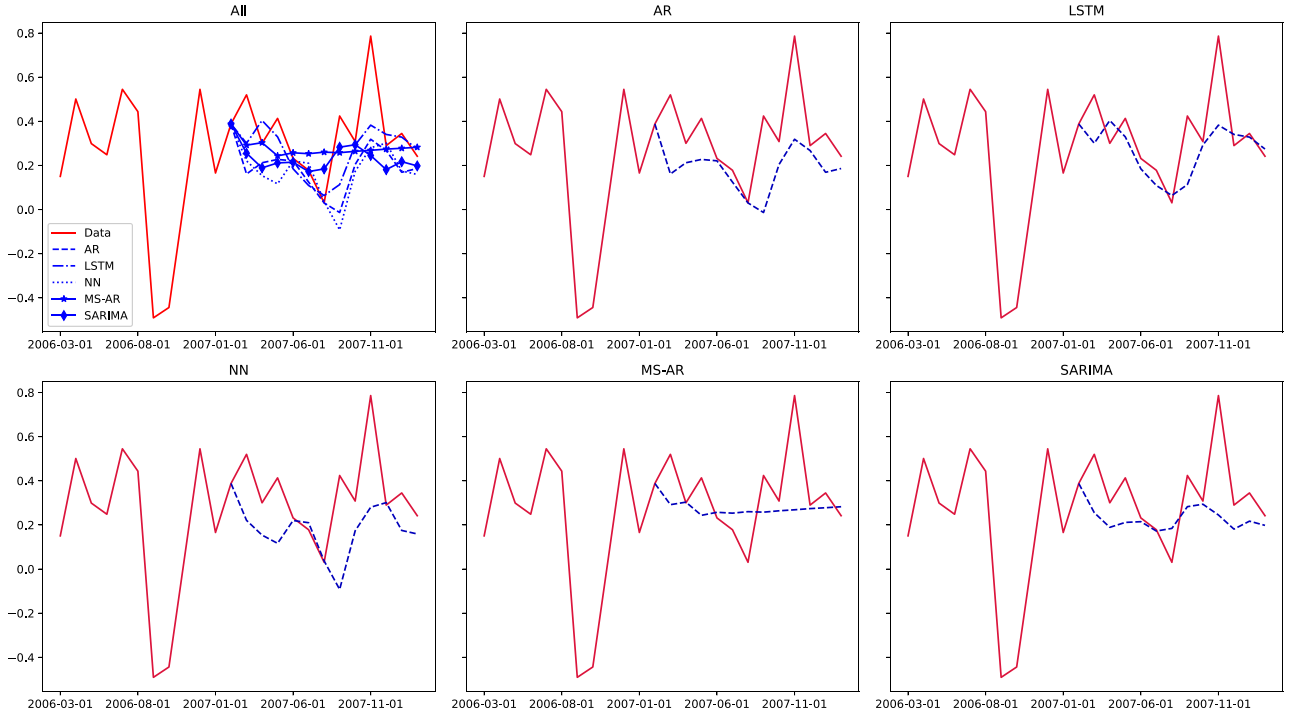

FIGURE 8 Real-time forecast 2007-03

predict that inflation would stay close to an extreme value for an extended period of time.

Robustness Checks. The results presented in this section might be affected by the choice of data or methods. We repeated the analysis on the annual inflation data and gained analogous results. We also tried to use forecast trimming as in Stock and Watson (1998) and replaced the forecast that induced a change in inflation larger than ever observed before by a no-change forecast. This did not affect the results. We also experimented with using an ensemble of models with different hyperparameters for NN and LSTM. This did not change the results either because as it will be shown in the next session, the hyperparameters do not significantly affect the performance of these network models. We also tried to vary the specification of the SARIMA model, the one presented here is the model version that delivers the best forecasting performance. Results for non-regularized AR

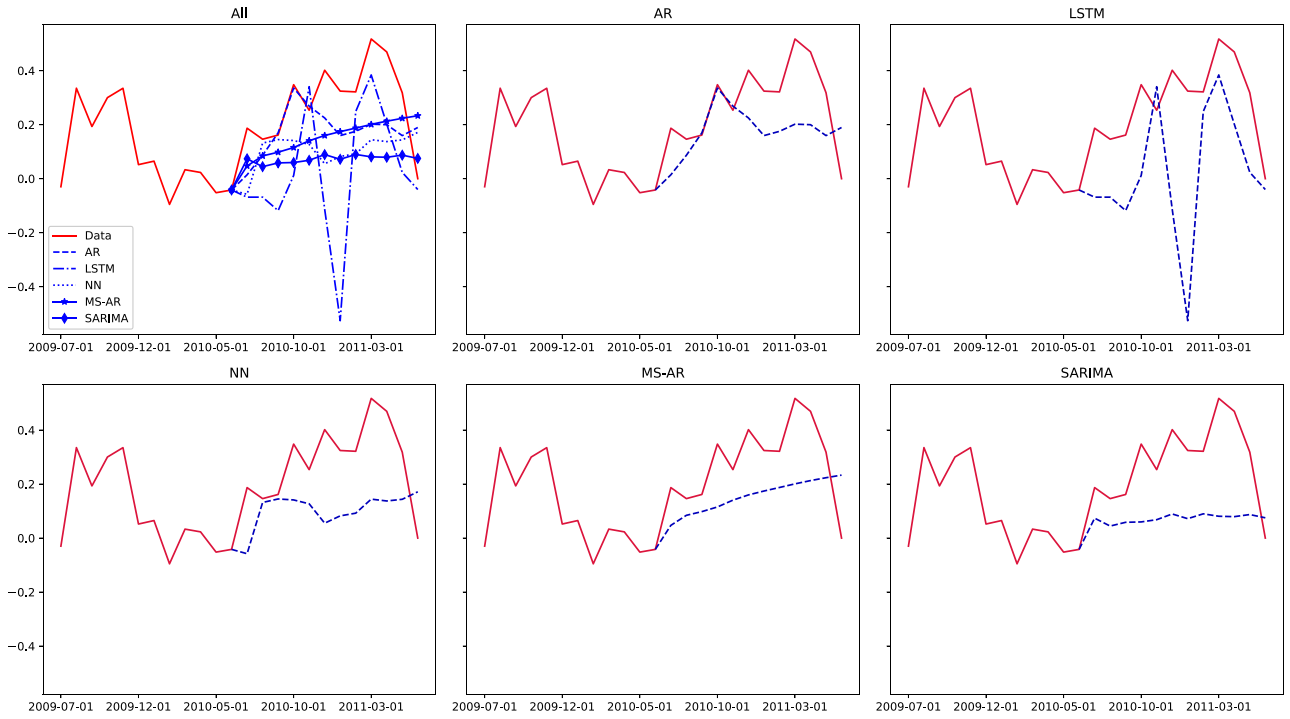

FIGURE 9 Real-time forecast 2010-07

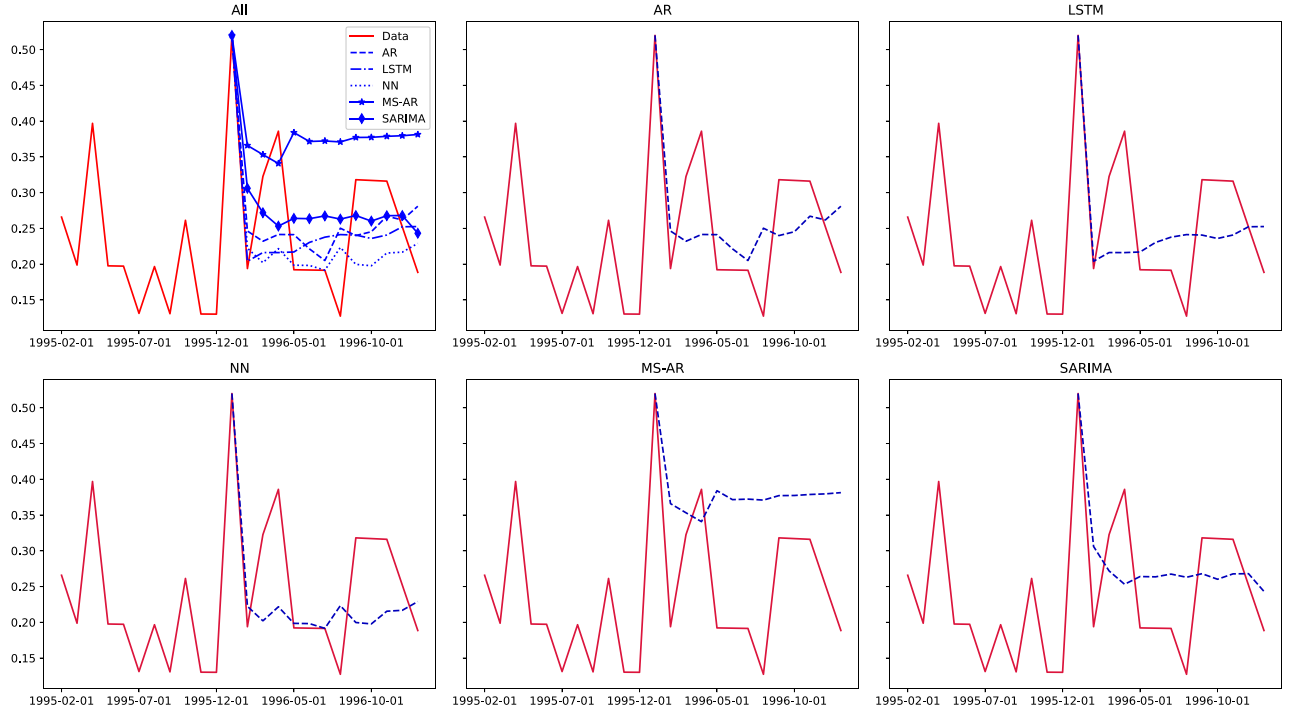

FIGURE 10 Real-time forecast 1996-02

models, trained with OLS were insignificantly different from regularized AR models reported in the tables above. Finally, we evaluated the sensitivity of the forecasting performance of the neural network models—NN, AR, and LSTM—to the random initialization of the parameters before training and concluded that random initialization does not affect the performance on the test set.

4 | MODEL SELECTION AND INTERPRETATION

This section presents knowledge about general neural network model performance, convergence characteristics, the impact of random initialization (Section 4.1), parameter choice (Section 4.2), and interpretation of trained models (Section 4.3), which was gained in this work.

4.1 | Initialization and convergence

Because our networks are trained by a local optimizer, multiple optima might be a concern. For example, the loss function of the used LSTM model is high-dimensional and highly nonlinear. It contains 40901[10] parameters and is hard to optimize.

We followed the literature of Stock and Watson (1998) and looked at the distributions of the forecast errors after training the networks on the same training and estimating on the same validation sets but with randomly chosen initial values for the parameters. This exercise did not show any significant variance with regard to the random initialization. Low variance of the attained performance in this random initialization experiment indicates that the models converge to approximately equal forecast in terms of RMSFE in each optimization chain.[11]

In a similar exercise, we randomly draw different validation sets together with random initialization. In this case, standard deviations of the RMSFE increased, but their value remained one order of magnitude below the absolute value of RMSFE itself. Network performance, thus, varies slightly over the dataset.

TABLE 3 Top best models of AR with parameters

	Test error	$h=2$	$h=3$	$h=6$	$h=12$	infc	p	Lag	LR	Epochs
1	0.20	0.22	0.22	0.23	0.25	None	24	24	0.003	500
2	0.20	0.24	0.26	0.28	0.27	bic	12	24	0.001	10000
3	0.21	0.22	0.23	0.24	0.26	None	24	24	0.100	18000
4	0.21	0.21	0.23	0.22	0.24	None	24	24	0.010	9000
5	0.21	0.22	0.23	0.23	0.25	None	24	24	0.300	1500
6	0.21	0.24	0.24	0.30	0.27	bic	12	24	0.050	9000
7	0.21	0.23	0.26	0.29	0.28	bic	12	12	0.003	1500
8	0.21	0.23	0.22	0.23	0.25	None	24	24	0.100	14000
9	0.21	0.20	0.22	0.23	0.26	None	24	24	0.050	1000
10	0.21	0.21	0.21	0.23	0.24	None	24	24	0.050	16000

Note: Selection is based on the average one-step-ahead RMSFE on SA data. h—number of forecast steps; infc—information criterion; p is either the optimally selected number of lags or the max lag; Lag—maximum number of lags.

TABLE 4 Top best models of NN with parameters

	Test error	$h=2$	$h=3$	$h=6$	$h=12$	n	infc	p	Lag	LR	Epochs
1	0.21	0.23	0.24	0.28	0.28	20	bic	12	12	0.001	500
2	0.21	0.23	0.26	0.28	0.29	10	bic	12	12	0.001	500
3	0.22	0.23	0.25	0.26	0.28	10	hqic	12	12	0.001	1500
4	0.22	0.23	0.24	0.29	0.27	10	hqic	12	12	0.001	1000
5	0.22	0.23	0.25	0.29	0.29	10	aic	12	12	0.001	1000
6	0.22	0.23	0.23	0.27	0.27	50	bic	12	12	0.001	500
7	0.22	0.21	0.24	0.30	0.28	50	bic	12	24	0.001	500
8	0.22	0.22	0.21	0.24	0.25	50	hqic	15	24	0.001	500
9	0.22	0.22	0.23	0.30	0.28	50	None	12	12	0.001	500
10	0.22	0.20	0.20	0.22	0.25	10	None	24	24	0.003	500

Note: Selection is based on the average one-step-ahead RMSFE on SA data. h—number of forecast steps; n—number of hidden units; infc—information criterion; p is either the optimally selected number of lags or the max lag; L—maximum number of lags; LR—learning rate.

4.2 | Sensitivity analysis

The performance of neural networks often depends crucially on the selected hyperparameters. It can also depend on the lag selection criterion and maximum number of lags that this criterion is allowed to choose from. To access the sensitivity of our forecasting models to the parameter choices, we take a closer look at the hyperparameter sets, which resulted in the lowest forecast error for each model type.

Tables 3–5 present the top 10 performers among AR models, top 10 among NNs, and top 10 among LSTMs,

TABLE 5 Top best models of LSTM with parameters

	Test error	$h=2$	$h=3$	$h=6$	$h=12$	n	infc	p	Lag	LR	Epochs
1	0.17	0.20	0.18	0.21	0.22	50	None	24	24	0.001	2000
2	0.17	0.21	0.21	0.25	0.26	20	None	24	24	0.001	1500
3	0.18	0.21	0.21	0.24	0.23	100	None	24	24	0.300	500
4	0.18	0.22	0.21	0.24	0.25	50	None	24	24	0.050	1000
5	0.18	0.19	0.20	0.22	0.24	50	None	24	24	0.010	2000
6	0.18	0.20	0.22	0.22	0.24	50	None	24	24	0.100	1000
7	0.18	0.20	0.20	0.21	0.25	100	None	24	24	0.050	2000
8	0.18	0.20	0.20	0.21	0.24	50	None	24	24	0.050	1500
9	0.19	0.20	0.20	0.21	0.22	100	None	24	24	0.300	1500
10	0.19	0.21	0.21	0.24	0.23	20	None	24	24	0.050	1500

Note: Selection is based on the average one-step-ahead RMSFE on SA data. h—number of forecast steps; n—number of hidden units; infc—information criterion; p is either the optimally selected number of lags or the max lag; L—maximum number of lags; LR—learning rate.

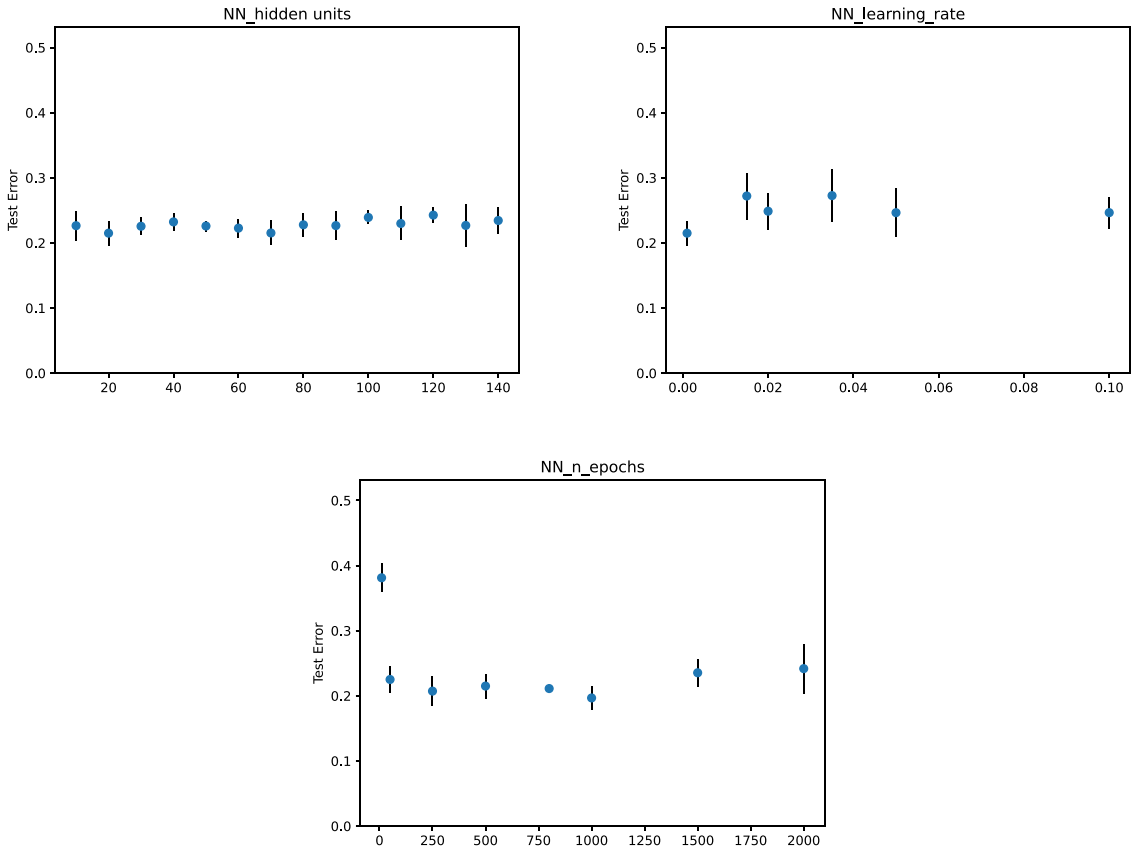

FIGURE 11 Sensitivity analysis, simple NN. Dots indicate mean RMSFE and black bars indicate 90% credible sets after 20 runs of Monte-Carlo cross-validation

respectively, as measured by the RMSFE for seasonally adjusted data. The ranking of their performance was done after cross-validation over 10 random splits between training and validation data. Note that this training procedure differs from the earlier real-time exercise, where a cross-validation would not be possible since the test set is fixed for each step. For the results in this section, the validation data consists of a few randomly chosen nonoverlapping intervals within the training dataset.[12]

All models are very close in terms of their validation errors and the exact ranking is not highly informative. Another observation is that the top 10 AR and top 10 NN models perform very similarly to each other while the performance of the LSTM is affected slightly more strongly by the selected hyperparameters.

Figures 11 and 12 present the sensitivity of the best performing NN and LSTM models, respectively. To create these plots, we took the top performing NN or LSTM model, varied one hyperparameter of interest at a time—the number of hidden units, initial learning rate for the optimizer, or the number of training epochs—and documented the changes in the model's performance.

Based on these plots and the tables above, we now mention some recommendations regarding the hyperparameters for each model type.

AR model. The AR model performs well, when the number of lags is not selected by information criteria, but instead is fixed at a maximum number. Although the Bayesian information criterion (BIC) can be a good choice as well, based on the results in Table 3. A wide variety of learning rate choices lead to equally performing well. The same is true for the number of training epochs—as long as it is at least 500.

NN model. As can be seen in Table 4 the simple NN performs best when Bayesian information criterion is used for the lag length selection and when the maximum number of lags to select from is large. The performance of the NN is insensitive to the number of hidden units, but has a bit more stable results when the number is below 80. Initial learning rate is also mostly irrelevant for the NN's performance. Best results are achieved with smaller learning rates below 0.05. The bottom plot on the Figure 11 indicates that the test error of the NN achieves minimum at around 1000 epochs and slightly increases afterwards.[13]

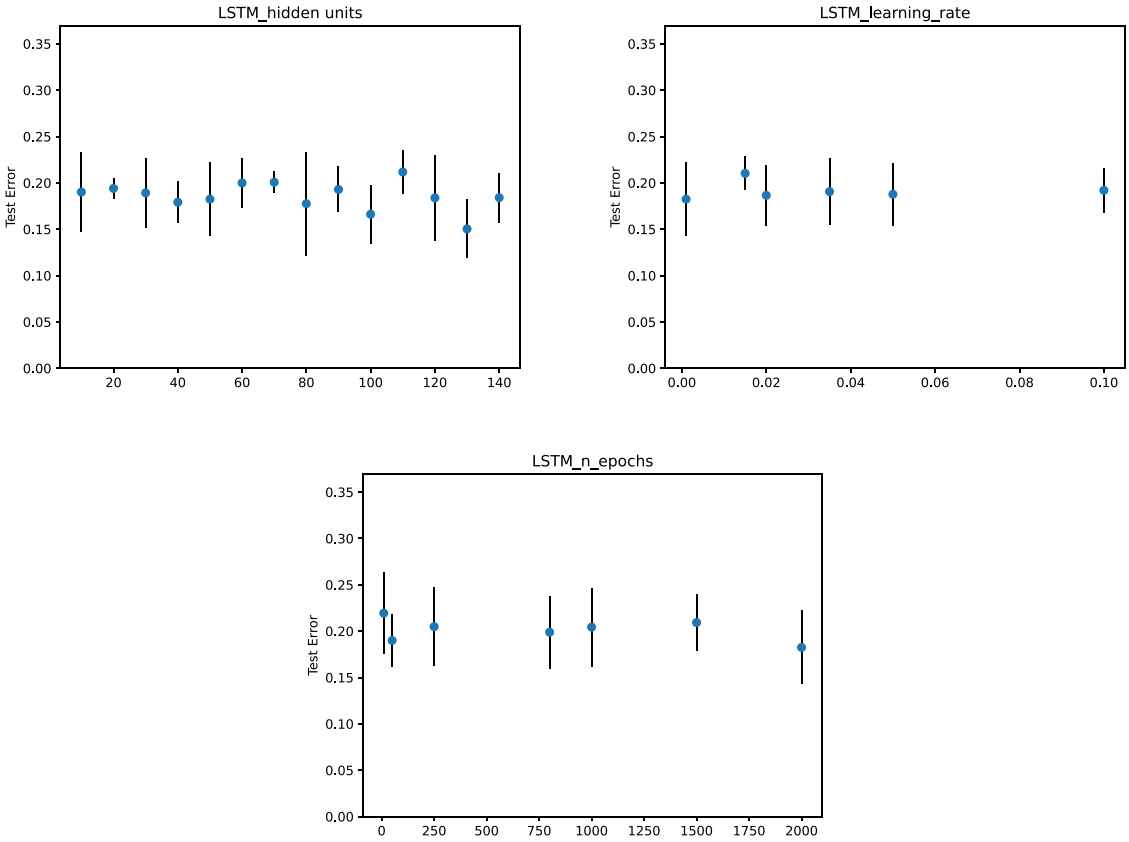

FIGURE 12 Sensitivity analysis, LSTM. Dots indicate mean RMSFE and black bars indicate 90% credible sets after 20 runs of Monte-Carlo cross-validation

LSTM model. The results for the LSTM suggest that the information criteria cannot improve performance. Instead, the maximum number of input lags should be given (we tested a maximum of 24). The forecast error is hardly impacted by the number of hidden units. Already as few as 20 can give peak results. As for the NN, the LSTM results are highly insensitive to the learning rate parameter and for both models it is essential to allow for a reasonable number of the training epochs. The best LSTMs are normally trained for 1000–2000 epochs, whereas the biggest gain in performance is already done within the first 200 epochs.

In summary, a researcher should bear in mind two aspects when deciding on the neural networks architecture: First, the maximum number of lags for the information criterion should not be too small; second, the researcher should train the model for a sufficient amount of time. Other hyperparameters appear not to be much of a concern for the task of inflation forecasting. This findings confirm the general conclusion in Bengio (2012).

4.3 | Understanding the mechanism: Layer-wise relevance propagation

After fitting NN and LSTM models, one is naturally interested in interpreting what the networks learned from the data and understanding what features of the input (in our applications different lags) are the most important for the network prediction. In contrast to linear models, it is not possible to directly interpret the weights of the neural networks because the final prediction is a nonlinear function of the network parameters.

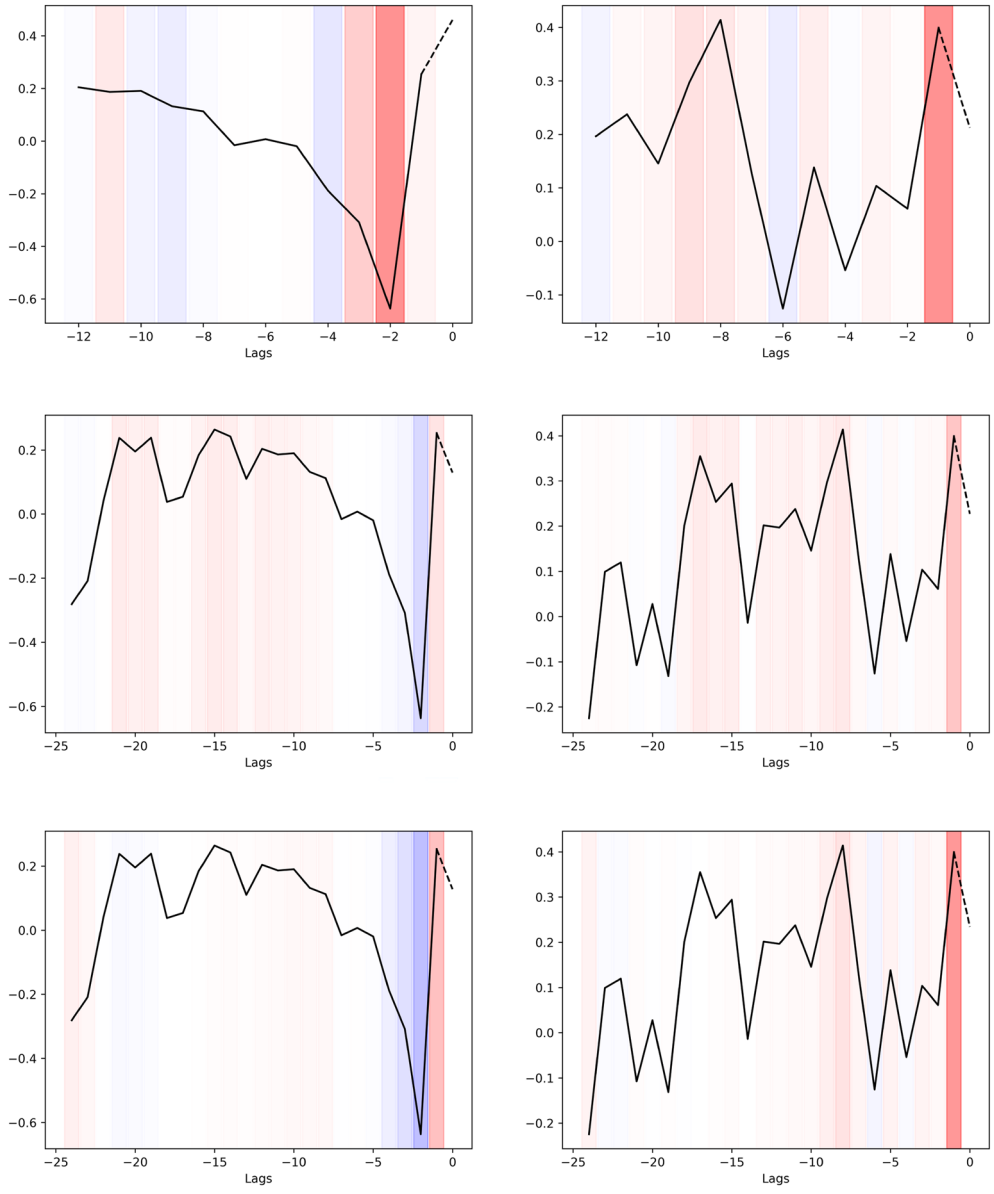

FIGURE 13 LRP plots for NN (top row), LSTM (middle), and AR (bottom row) for SA data

Lapuschkin et al. (2016) suggest to access the importance of model inputs by layer-wise relevance propagation (LRP). A detailed description of the LRP algorithm lies beyond the scope of this paper and we will only sketch the main idea. The LRP procedure attaches a value to every neuron (including the input neurons—data lags in our case) that quantifies its contribution to the network output. This value is called "relevance." It is computed layer-wise by aggregating the signals that a neuron contributes to its successors. The signal's value depends on the weights attached to the connections between a given neuron and its successors. Relevance of the input "neurons" give an estimation of their relative importance for the final prediction.

Figures 13 and 14 each depict two examples of the LRP measurement for the top performing NN (top row), top performing LSTM (middle row), and top-performing AR model (bottom row) for seasonally adjusted and non-adjusted data, respectively. These were chosen as representative examples and illustrate, what lags the methods pay attention to given an input sequence. The black solid line depicts the actual data. Each bar represents the relevance of a particular lag, which was fed into the model as an input. The LSTM has 24 lags and the NN has 12. The rightmost value of each plot depicted with a dashed line is the one-step-ahead prediction of the corresponding network. Red bars indicate the lags that contribute positively to the predicted values and blue bars imply that the value at the corresponding lag tells the network to decrease the final prediction. Intensity of the color measures the magnitude of the relevance so white bars represent zero relevance.

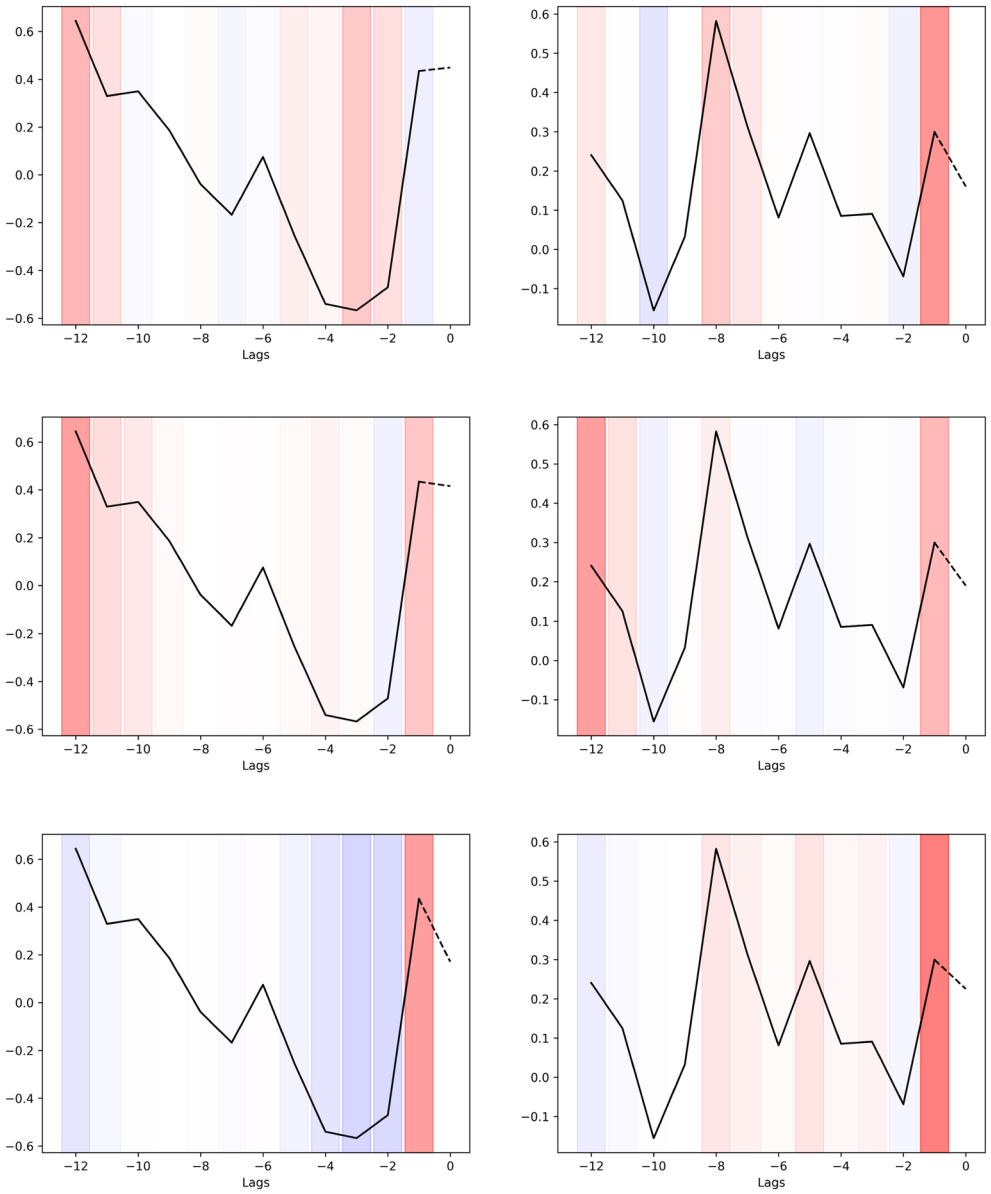

FIGURE 14 LRP plots for NN (top row), LSTM (middle), and AR (bottom row) for NA data

Several observations are worth pointing out. Very common in these plots is, that the most recent lags contribute most strongly to the result, that is, have the highest relevance. The sign and weight of this contribution can differ between models even on the same input.

For seasonally adjusted data, LRP-plots for LSTM and AR look very similar. These models make similar decisions, which is reflected in their performance. LSTM and AR models are especially sensitive to recent lags, that contribute positively (red) if they correspond to positive values of inflation and negatively (blue) if they correspond to negative values. The NN model on the other hand looks for and reacts to extrema (minima and maxima) in the input.

For non-adjusted data, the LSTM model seems to learn a pattern that is closer to what the NN model learns. The LRP-plot of the LSTM is now very different from the one for the AR. The AR model learns that the most recent lag contributes rather positively, lags further into the past contribute less and/or negatively. Because the AR model does not know about the time component of the data, it simply gives the highest weight to the most recent lag.

In contrast to the AR, for the LSTM model the very recent lag as well as the 10–12th lags are important. The LSTM network realizes that the 12th lag contains seasonal information, which is important for the prediction. Additionally, the LSTM learns a flexible pattern for the remaining lags, where positive and negative relevance alternates.

In summary, the nonadjusted data lead to the LSTM and NN models attributing relevance to three parts of the input: the most recent lags, the lags one year ago and the lags of extreme values in the input sequence. This last part allows to adjust a forecasted value if some extreme turns in the trend were observed. It could be interpreted as a combination of nonlinearity and a "nonsparsity" of the model in a sense that the LSTM pays equal attention to all data lags. This conclusion is in accordance with Medeiros et al. (2018), who argue that machine learning methods select non-sparse model specifications in a multivariate forecasting set-up. We can also understand why performance of the LSTM is closer to the SARIMA on nonseasonally adjusted data. It turns out, that LSTM learns a seasonal pattern in the data without being explicitly constructed to do so.

5 | CONCLUSION

This paper evaluates the performance of a nonlinear machine learning method—the long short-term memory recurrent neural network (LSTM)—for univariate inflation forecasting and compares it to the standard fully connected neural network as well as to classical methods—the random walk model, linear autoregression models, seasonal ARIMA model, and the Markov switching model. Performance was tested in a real-time forecasting exercise by computing predictions in a rolling-window setting.

Our findings suggest that if one is using neural networks for prediction of economic time-series, an additional improvement in performance can be achieved by using a particular type of neural network that is designed for sequential data. We show that LSTM is an efficient method for inflation prediction that allows to improve forecasting accuracy upon other all other classical and machine learning models except SARIMA on the non seasonally adjusted data. For seasonally adjusted data our results are mixed. We find that all models perform very similar to each other and the exact ranking is unstable across different starting dates of the forecast and seasonal filters.

Our layer-wise relevance propagation analysis suggests that LSTM is able to outperform simple NN and regularized AR models because it takes into account the entire lag structure given as an input. It pays attention especially to the highest and lowest values of the lagged inflation—that is, turning points. Simple NN and AR models, in contrast, focus mostly on recent lags and, in the case of nonadjusted data, on the lags 1 year ago. We also noted that LSTM is more efficient in predicting the mean-reversion of the time-series after turning points than SARIMA or MS-AR. The performance of the LSTM is on par with the seasonal SARIMA on nonseasonally adjusted data because, as we show with the help of the layer-wise relevance propagation technique, the LSTM learns something very close to a seasonal pattern without being explicitly instructed to do so. It is perhaps not surprising that seasonal variation drives most of the results in a univariate forecasting exercise. Performance of LSTMs in computation of multivariate forecasts would be a natural next question that we hope to further explore in future research. It also remains an open question whether the efficiency of long-term forecasts can be further improved by explicitly optimizing the models for a particular forecast horizon (direct forecasts) or for an average inflation rate over the next 12 months. We leave this question for our future projects as well.

We additionally show that the LSTM models are rather insensitive to hyperparameter choices in the inflation prediction task. The number of hidden units and the number of training epochs should not be chosen too small—apart from that performance does not vary greatly when changing different hyperparameters.

We want to stress that our goal is not to compare LSTM with all models that can potentially be used for inflation prediction in a large-scale "horse race" exercise. We believe that any forecaster makes her judgment based on several models anyway. We thus suggest that the performance of the LSTM is on par or better than of some standard forecasting models and that the LSTM should thus be included in the ensemble of methods applied.

ACKNOWLEDGMENTS

We thank Mark Watson, Chris Sims, Grégoire Montavon, Danilo Cascaldi-Garcia, Michael Burda, Alex Meyer-Gohde, Andreas Tryphonides, three anonymous referees and participants in several conferences and seminars for comments and suggestions. Anna Almosova's work on this paper was partially supported by Schwerpunktprogramm 1764 of the German Science Foundation. Open Access funding enabled and organized by Projekt DEAL.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are openly available in the FRED database of the Federal Reserve Bank of St. Louis. It is accessible at https://fred.stlouisfed.org, reference numbers CPALTT01USM661S (for seasonally adjusted monthly data), and CPALTT01USM657N (for nonseasonally adjusted monthly data).

ORCID

Anna Almosova https://orcid.org/0000-0002-9606-3254

ENDNOTES

[1] For example Stock and Watson (2007) fit an integrated moving average (time-varying trend-cycle) model to the GDP-deflator data, and show that the coefficients in this model changed in the beginning of 70s and then again in the mid 80s. The authors conclude that "...if the inflation process has changed in the past, it could change again."

[2] We use Python TensorFlow on 4 NVIDIA K20m GPUs.

[3] In practice it means that the data is available at monthly frequency and for an extended period of time.

[4] MS-AR and SARIMA models have hyperparameters as well, however, much fewer.

[5] All the models are fit under the early-stopping rule (see Section 2.3) for a better generalization. As a result, the estimators are biased by construction in any case.

[6] We use rectified linear unit (ReLU), which is defined as $\sigma(x) = \max(0, x)$. Our choice is motivated by the computational efficiency of this nonlinear function (Nair & Hinton, 2010).

[7] Available on https://www.statsmodels.org/stable/index.html.

[8] Alternatively, one can use a Bayesian AR model with a prior that imposes shrinkage.

[9] The overall magnitude of the MSEs lies in the ballpark that is reported in the literature. For example, Stock and Watson (1999) find that a univariate RW has an MSE of 0.39% and the MSE of a univariate AR model is 0.16% when forecasting inflation 12 months ahead for for 1984–1996. In our analysis for 1960-2020 this errors are respectively 0.35 and 0.11 for NA data, and 0.34 and 0.08 SA data.

[10] This LSTM model has 100 hidden units, at each time step each of its 4 gates transforms 100 states from the previous step and a 1-dimensional input into 100 updated states. Each update also contains a bias parameter. The final state of the network is transformed into the final 1-dimensional output by multiplying the state by a (100x1) matrix. A bias is added at this final step. As a result the LSTM has $4 \times ((101 \times 100) + 100) + 100 + 1 = 40901$ parameters.

[11] This, of course, does not guarantee that models with different initializations would attain an equal performance in real-life applications, see (D'Amour et al., 2020) for a discussion.

[12] Consequently, the cross-validated test errors imply lower MSFE than shown in the Table 2.

[13] Note that the training error is strictly decreasing in the number of training epochs while the test error is not—the problem called overfitting. Training of the networks can therefore be stopped at the point of the test error minimum and before the train error reaches its minimum. This represents the "early-stopping" principle.

REFERENCES

Ahmed, N.K., Atiya, A.F., Gayar, N.E., & El-Shishiny, H. (2010). An empirical comparison of machine learning models for time series forecasting. *Econometric Reviews*, *29*(5-6), 594–621.

Arras, L., Montavon, G., Müller, K.-R., & Samek, W. (2017). Explaining recurrent neural network predictions in sentiment analysis. arXiv preprint arXiv:1706.07206.

Atkeson, A., & Ohanian, L.E. (2001). Are Phillips curves useful for forecasting inflation? *Quarterly Review*, *25*(1), 2–11. https://ideas.repec.org/a/fip/fedmqr/y2001iwinp2-11nv.25no.1.html

Barron, A.R. (1993). Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information theory*, *39*(3), 930–945.

Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures, *Neural networks: Tricks of the trade*: Springer, pp. 437–478.

Chen, X., Racine, J., & Swanson, N.R. (2001). Semiparametric ARX neural-network models with an application to forecasting inflation. *IEEE Transactions on neural networks*, *12*(4), 674–683.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, *2*(4), 303–314.

D'Amour, A., Heller, K., Moldovan, D., Adlam, B., Alipanahi, B., Beutel, A., Chen, C., Deaton, J., Eisenstein, J., Hoffman, M. D., & Hormozdiari, F. (2020). Underspecification presents challenges for credibility in modern machine learning. arXiv preprint arXiv:2011.03395.

Elger, T., Binner, J., Nilsson, B., & Tepper, J. (2006). Predictable non-linearities in US inflation. *Economics Letters*, *93*, 323–328.

Grigsby, J., Wang, Z., & Qi, Y. (2021). Long-range transformers for dynamic spatiotemporal forecasting. arXiv preprint arXiv:2109.12218.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Kingma, D.P., & Ba, J. (2014). Adam: A method for stochastic optimization. http://arxiv.org/abs/1412.6980

Kock, A.B., & Teräsvirta, T. (2011). Forecasting macroeconomic variables using neural network models and three automated model selection techniques. (*Technical Report*): Department of Economics and Business Economics, Aarhus University.

Kuan, C.-M., & Liu, T. (1995). Forecasting exchange rates using feedforward and recurrent neural networks. *Journal of Applied Econometrics*, *10*(4), 347–364.

Lapuschkin, S., Binder, A., Montavon, G., Müller, K.-R., & Samek, W. (2016). The LRP toolbox for artificial neural networks. *The Journal of Machine Learning Research*, *17*(1), 3938–3942.

Lim, B., Arık, S.Ö., Loeff, N., & Pfister, T. (2021). Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, *37*(4), 1748–1764. https://www.sciencedirect.com/science/article/pii/S0169207021000637

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

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018). Statistical and machine learning forecasting methods: Concerns and ways forward. *PloS one*, *13*(3), e0194889.

Marcellino, M., Stock, J.H., & Watson, M.W. (2006). A comparison of direct and iterated multistep ar methods for forecasting macroeconomic time series. *Journal of Econometrics*, *135*(1-2), 499–526.

McAdam, P., & McNelis, P. (2005). Forecasting inflation with thick models and neural networks. *Economic Modelling*, *22*(5), 848–867.

Medeiros, M.C., Vasconcelos, G., Veiga, A., & Zilberman, E. (2018). Forecasting inflation in a data-rich environment: The benefits of machine learning methods. SSRN working paper.

Nair, V., & Hinton, G.E. (2010). Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 807–814.

Nakamura, E. (2005). Inflation forecasting using a neural network. *Economics Letters*, *86*(3), 373–378.

Salinas, D., Flunkert, V., Gasthaus, J., & Januschowski, T. (2020). Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, *36*(3), 1181–1191. https://www.sciencedirect.com/science/article/pii/S0169207019301888

Smalter Hall, A., & Cook, T.R. (2017). Macroeconomic indicator forecasting with deep neural networks. (*Working Paper 17-11*): Federal Reserve Bank of Kansas City.

Stock, J.H., & Watson, M.W. (1998). A comparison of linear and nonlinear univariate models for forecasting macroeconomic time series. (*Working Paper 6607*): National Bureau of Economic Research. http://www.nber.org/papers/w6607

Stock, J.H., & Watson, M.W. (1999). Forecasting inflation. *Journal of Monetary Economics*, *44*(2), 293–335.

Stock, J.H., & Watson, M.W. (2007). Why has us inflation become harder to forecast? *Journal of Money, Credit and Banking*, *39*, 3–33.

Swanson, N.R., & White, H. (1997a). Forecasting economic time series using flexible versus fixed specification and linear versus nonlinear econometric models. *International Journal of Forecasting*, *13*(4), 439–461.

Swanson, N.R., & White, H. (1997b). A model selection approach to real-time macroeconomic forecasting using linear models and artificial neural networks. *Review of Economics and Statistics*, *79*(4), 540–550.

Teräsvirta, T. (2006). Forecasting economic variables with nonlinear models. *Handbook of economic forecasting*, *1*, 413–457.

Teräsvirta, T., & Case, H. (2017). Nonlinear models in macroeconometrics, *Oxford research encyclopedia in economics and finance*: Oxford University Press.

Teräsvirta, T., Van Dijk, D., & Medeiros, M.C. (2005). Linear models, smooth transition autoregressions, and neural networks for forecasting macroeconomic time series: A re-examination. *International Journal of Forecasting*, *21*(4), 755–774.

AUTHOR BIOGRAPHIES

Anna Almosova is a junior professor in Economics at Technical University Berlin and Einstein Center for Digital Future since October 2019. She has a PhD in Economics from Humboldt University Berlin and is working on applied Bayesian econometrics, machine learning and cryptocurrencies. She has two master's degrees in Economics: From Humboldt University Berlin and from Higher School of Economics, Moscow. During her graduate studies Anna was a visiting graduate student at Princeton University (invited by Chris Sims), worked as an intern at the International Monetary Fund in Washington DC and completed a research internship in the German Ministry for Economic Affairs and Energy in Berlin

Niek Andresen is a PhD student at Technische Universität Berlin, where he started in August 2019 in the Science of Intelligence Cluster of Excellence. He studies Computer Vision and Artificial Intelligence using Neural Networks and observation of natural intelligence. Other scientific interests include econometrics, 3D reconstruction and computational biology.

How to cite this article: Almosova, A., & Andresen, N. (2023). Nonlinear inflation forecasting with recurrent neural networks. *Journal of Forecasting*, *42*(2), 240–259. https://doi.org/10.1002/for.2901

APPENDIX A: LSTM CELL

The LSTM cell at time t receives three input vectors: the cell state c_{t-1}, the hidden state h_{t-1} and the new lag of the data y_{t-1}. The cell state is altered at each time step and then handed over to the next one. The two alterations to it are computed through small fully connected neural networks taking the concatenation of the hidden state and the current lag as input. The first adjustment to the cell state is a forget operation, where every value of c_{t-1} is multiplied by a number between 0 (forgetting the information) and 1 (keeping the value entirely). This number is computed by the σ-gate, that is depicted on the left of Figure 6. It is a small neural network, that has a sigmoid function as activation at the output. The second adjustment to the cell state follows thereafter. It is the addition of new information to c_{t-1}. To every of its values, a number between -1 and 1 is added. These numbers are computed by first running h_{t-1} and y_{t-1} through a small neural network having a $\tanh$ function as activation at the output. These transformed values are then multiplied by the output of another σ-gate, which again has the ability to forget some of the newly computed information. After the forget and add operations, the new cell state c_t is complete and is handed to the next time step. Finally, the updated hidden state is computed. This is done by transforming the cell state with a $\tanh$ activation and forgetting parts of it again through another σ-gate having once again the previous h_{t-1} and y_{t-1} as input. The final prediction $\hat{y}_{t|t-1}$ is a copy of the last h_t.