Nonlinear Inflation Forecasting with Recurrent Neural Networks

Latest version is available here

Anna Almosova*, Niek Andresen†

May 2019

This paper demonstrates the value of nonlinear machine learning techniques in forecasting macroeconomic time series. We show that a long short-term memory (LSTM) recurrent neural network outperforms the linear autoregressive model (AR), the random walk model (RW), seasonal autoregressive model (SARIMA), Markov switching model (MS-AR) and the simple fully-connected neural network (NN) in forecasting monthly US CPI inflation. At all horizons the root mean squared forecast error (RMSFE) of the LSTM is approximately one third of the RMSFE of the random walk. We also show that rolling-window real-time forecasts of the LSTM are significantly more accurate compared to the AR and NN forecasts. Additionally we conduct a sensitivity analysis with respect to hyper-parameters and provide a qualitative interpretation of what the networks learn by applying a novel layer-wise relevance propagation technique. We conclude that the good performance of the LSTM results from their ability to capture nonlinearities in the data in a combination with their flexible architecture.

Keywords: Neural Networks, LSTM, Forecasting, Inflation
JEL classification: C45, C53, E37

*Humboldt University of Berlin, School of Business and Economics. Spandauerstr.1, 100178, Berlin, Germany. Email: anna.almosova.hu.berlin@gmail.com

†Technical University of Berlin, Department of Computer Engineering and Microelectronics. Marchstr.23, 10587 Berlin Germany. Email: andresen.niek@gmail.com

We thank Mark Watson, Chris Sims, Grégoire Montavon 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.

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, 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 & Watson (2007) point out, inflation in the US 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 & 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 (NNs) 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,

[1] For example Stock & 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."

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 RNNs significantly more feasible.

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 take monthly CPI inflation data for the US as a benchmark and present supplementary analysis for the PCE inflation in the Appendix.

One must note that the performance of neural networks is determined by several hyper-parameters such as the number of hidden units or the share of the data that is assigned as training set. To provide some guidance on how to choose these parameters we conduct an extensive sensitivity analysis. In total we train about 4300 different models. Moreover, neural networks are often criticized for their "black box" structure. Since 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.

According to our results, the LSTM outperforms all considered models at all forecasting horizons. The average improvement in terms of the root mean squared forecast error (RMSFE) is about 50-70%. One-step-ahead RMSFE of the all models is approximately 60% of the corresponding measure for the RW model. At longer horizons, 2 through 15 steps ahead, performance of the AR and NN deteriorates with the relative RMSFEs

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

rising up to about 90% of the random walk's error. Relative errors of the SARIMA model converge to one even faster; at 6 steps ahead SARIMA performance is on par with the RW and at 12-15 steps ahead it performs worse. The MS-AR model performs well at a 15 steps ahead horizon. However, its shorter forecasts are inferior to all other considered models. Finally, our main finding is that the LSTM produces the most accurate predictions at all horizons longer than 2 months. Its errors equal to approximately 50-30% of the random walk errors. Consequently, the main takeaway from out analysis is that the LSTM is an efficient nonlinear model for forecasting inflation especially at longer horizons. Our work suggests that long short-term memory recurrent neural networks in particular, and machine learning methods in general, deserve more attention from economists and policymakers.

To further support this statement we put ourselves in the world of policymakers who has to take decisions in real-time and computed real-time rolling-window forecasts from the year 2000 onward. This can be easily done with CPI data because it is not revised. We show that the LSTM achieves a better goodness of fit compared to the AR and the simple NN methods in terms of both the mean absolute forecast error and the mean squared forecast error. The difference in the accuracy of one-step-ahead monthly forecasts is significant as evaluated by the Deabold-Mariano test.

One might wonder how difficult it is to set up a proper neural network model given that these models are very rich in hyper-parameters. 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. Similarly, the performance of the LSTM model deteriorates if the number of lags to select from is too small. 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 he should

not include too few hidden units or allow for too few lags; second he should train the model for a sufficient amount of time. Other hyper-parameters appear not to be much of a concern.

Rather then treating the neural networks as "black box" models we try to shed some light on the reasons behind their good performance. We conduct the layer-wise relevance propagation LRP analysis, that allows us to identify which model inputs, that is past lags of inflation, are the most important for the prediction of the inflation today. For example, the simple NNs mostly pay attention to the most recent lag and the same lag one year ago when computing their predictions. It seems like neural networks behave similarly to the AR(1) model. Additionally, they are able to detect the annual seasonality in the data and take this into account. For the LSTMs the picture is strikingly different. The pattern it learns is more complex and differs significantly for different data points. We conclude that the LSTM performance results from its ability to explore the nonlinearity in the inflation dynamics and its flexibility to adjust the importance of different input lags for the final predicted value depending on the sample.

This paper is not alone in applying deep learning methods to macroeconomic forecasting. Ahmed *et al.* (2010) and Stock & 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 & Liu (1995) who demonstrated the success of simple and recurrent NNs for the exchange rate forecasting in several countries, Swanson & White (1997b,a) who evaluated the advantage of the NNs with time varying coefficients in a real-time forecasting setup, Kock *et al.* (2011) who discuss direct versus indirect forecasts with NNs.

Chen *et al.* (2001), Nakamura (2005) and McAdam & 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 RNN 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 & Watson (1999) found that at 12 months ahead horizon nonlinear methods perform the best for some macro-variables, 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 hyper-parameters 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), Teräsvirta & 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 and the current paper supports this conclusion by presenting the evidence for univariate recurrent neural networks.

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 lays out the sensitivity analysis and section 5 presents the layer-wise relevance propagation analysis. Section 6 concludes.

2. Methodology

We rank the forecasting methods based on the root mean squared forecast error (RMSFE) for out-of-sample forecasts on the test set. The share of the test set is set to 10% of the whole data sample, that is the number of observations in the test set N^{test} is equal to 10% of the total number of observations. Test sets are constructed by randomly drawing N^{test} non-overlapping data sequences of the length p in a Monte Carlo cross validation setting, where p is the selected lag length . The same procedure is applied to obtain validation sets for the sensitivity analysis and real-time forecasts. Note that since our test set is randomly drawn from the entire dataset, our cross-validation results converge to the leave-one-out cross-validation (LOOCV). However, we can set the number of replications to be smaller than the sample size as it is required for LOOCV to save computation time.

We consider indirect (rolling 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 *et al.* (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 forecasts which is made at date $t-1$. We start

from the following model:

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

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 information criterion.

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

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

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

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

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 2-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.[3] In other words, our nonlinear models receive solely the information about the first moment of the 1-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.

[3] All the models are fit under the early-stopping rule. The parameters are regularized in order to achieve a better generalization. As a result the estimators are biased by construction in any case.

2.1. Forecasting Models

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

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

The results are presented for the standard random walk model that is n=1. We also tried n=3, 6 or 12 and obtain very similar results in terms of model comparison.

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

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

3. NN: simple fully-connected neural network (Swanson & White, 1997b):

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

where $\sigma(\cdot)$ is a non-linear activation function,[4] 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 1 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 which is in turn a specific type a simple fully connected neural network (Figure 2). The difference between the NN and the recurrent neural network (RNN) comes from the recurrent nature of the latter. An RNN consists of a NN which is repeated as many times as there are data lags in the input. The prediction is computed sequentially after each time step of the input. Intermediate output, which is called the "state" of the network, is used as an additional input at the next time step. Representation of a standard recurrent neural network is given on the Figure 3. Recurrent propagation of the state is represented by the horizontal arrows. Note that the weights of the network stay the same, i.e. the "RNN" block in the schematic is identical for every time step.

[4] 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).

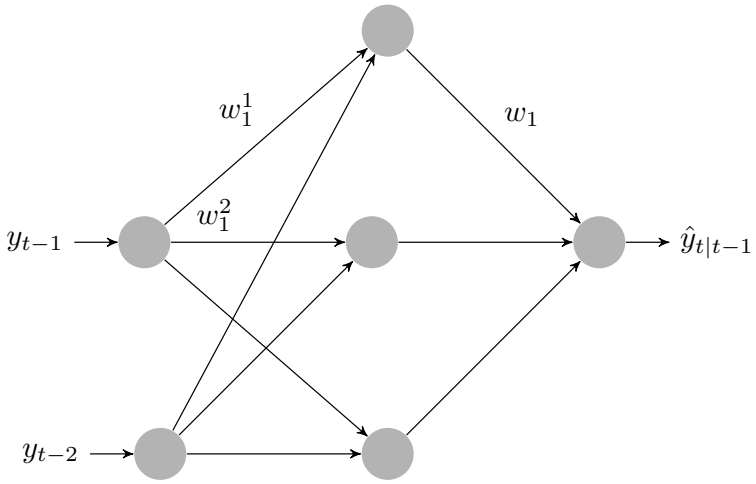

Figure 1: Fully-connected Neural Network (p=2)

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 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, i.e. p can be variable.

The LSTM network 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 4. This cell is used recursively as many times as there are data lags in the model.

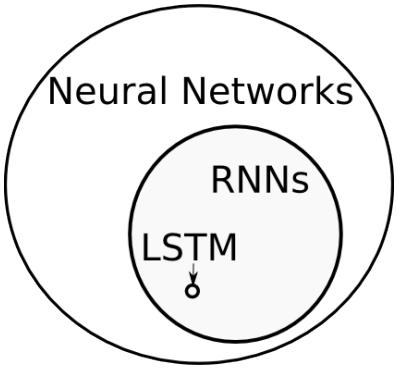

Figure 2: Classification of Artificial Neural Networks

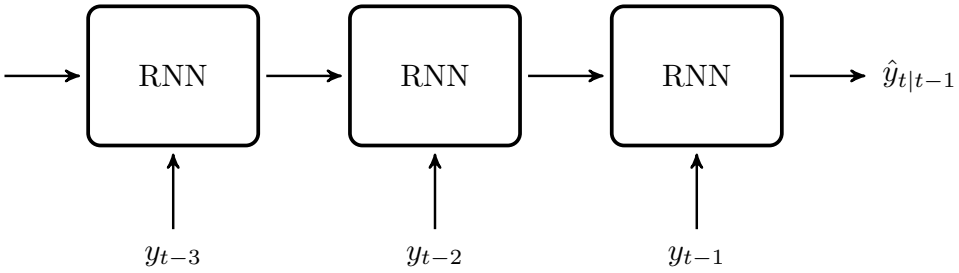

Figure 3: Basic Representation of a Recurrent Neural Network (p=3)

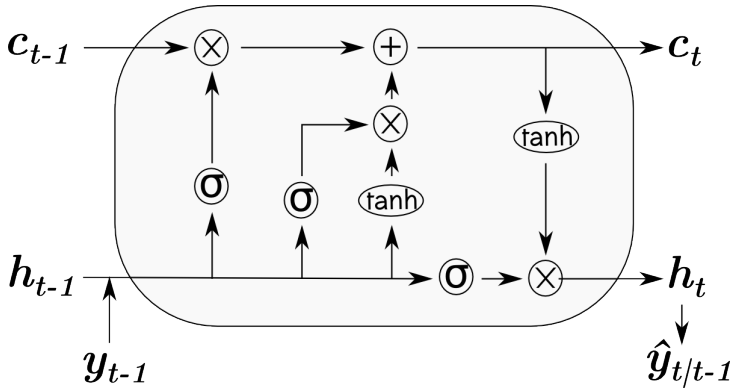

Figure 4: LSTM Cell Representation

Here y_{t-1} represents the input at the time step t (for example a lagged inflation value). c is the "state" of the network which represents its memory about the past. h_t denotes the output of the LSTM at the step t. σ and $tanh$ represent the gates which 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 that was computed on the previous step (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.

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[5] 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, that 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.[6] Note that the SARIMA and the MS-AR 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 structured. 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 hyper-parameters and local optima are applicable to SARIMA and MS-AR.

[5] Available upon request.

[6] Available on statsmodels.org

2.2. 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-oder 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: $\beta_1 = 0.9$, $\beta_2 = 0.999$.

Parameter updating is stopped when the RMSFE stops to decline for the test set and not for the train set. This early stopping rule regularizes the parameters and ensures a better generalization of the model. Since 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.[7] More specifically, we represent the AR model as a special case of a simple NN with a linear activation function and no hidden layer.

2.3. Data

We use monthly data on annual US CPI inflation from the FRED database of the Federal Reserve Bank of St. Louis for 1960:01 - 2018:07 which constitutes 703 observations (Appendix A presents the data series as well as the Dicky-Fuller test for non-stationarity).

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

Inflation rates are calculated as annual growth rates of the consumer price indices. The data are non-seasonally adjusted. Since we use annual growth rates the seasonality should not be an issue. Moreover, we do not to use the seasonally-adjusted series that are computed by the Bureau of Labor Statistics because the seasonality filters that they use can favor nonlinear methods. More specifically, the seasonal adjustment is a two-sided non-linear data transformation. Nonlinear methods can, potentially, achieve a good fit by simply unraveling the seasonal transformation. Additionally, the seasonally adjusted data are subject to annual revisions and thus cannot be used for our real-time forecasting exercise. We are also interested in testing the hypothesis that neural networks can automatically learn the seasonality of the data.

As a robustness check in the Appendix D we present the results for personal consumption expenditures (PCE) data for the same time span. PCE inflation rates were computed by the same procedure as for the CPI.

3. Results

3.1. Average Forecasting Performance

This section presents the results of the model estimation and compares their average forecasting performance on the test set. We start by accessing the accuracy of our models in terms of the average RMSFE of point forecasts, where the average is computed over 20 runs of Monte-Carlo cross-validation.

To rank the AR, NN and LSTM models we proceed as follows. On the first step for each model type we specify a range of possible values for every hyper-parameter. The hyper-parameters include information criteria (AIC, BIC, HQIC or use the given maximum lag length without any information criteria), maximum lag length to select from (12 or 24), number of hidden units for neural networks (range from 10 to 100) and the initial learning rate for the Adam optimizer (range from 0.001 to 0.3). In total we train 4420 different models. On the second step we select top 10% of each of the model types: AR, NN and LSTM. For those top candidates we run cross-validation to obtain mean and standard errors of the RMSFE at each forecast horizon. We finally rank the models based on their test errors after the cross-validation and present results for the

best performer for each model.

SARIMA and MS-AR models were estimated with the maximum likelihood method without a hyper-parameter tunning. Their errors are computed on the randomly sampled test sets exactly as for the other methods.

Table 1 summarizes the performance for all forecasting models relative to the RW which is taken as a benchmark. The results suggest 4 conclusions. First, at all horizons performance of all models is systematically superior to the simple RW model; the forecast errors of alternative models range between 23% and 90% of the RW's errors. Second, the LSTM gives more accurate predictions than all other methods even at the 15-step-ahead horizon. The RMSFE of the LSTM is on average just one third of the RW's. Third, performance of the AR and NN models track each other closely at each forecasting horizon. As the number of forecast steps h increases, the performance of the AR and NN models deteriorates and their forecast errors become closer in magnitude to the RW. Third, the performance of the SARIMA and the MS-AR models are in general slightly worse than of the other methods at all forecasting horizons. Our explanation for this observation is the different training procedure. The parameters of the AR, NN and LSTM models are regularized to achieve a better generalization. In contrast, the SARIMA and the MS-AR are estimated by the maximum likelihood.

Our results are in line with Elger *et al.* (2006) since we also find that the simple NN outperforms the MS-AR model at shorter horizons and becomes less accurate than the MS-AR at longer horizons (12 or more month ahead). However, in contrast to their results our specification of the recurrent neural network - the LSTM - outperforms the MS-AR model at all horizons.

Since our networks are trained by a local optimizer multiple optima might be a concern. We follow the literature (Stock & Watson, 1998) and look at the distributions of the forecast errors after training the networks on the same train and test set but with randomly chosen initial values for the parameters. Low variance in this random initialization experiment indicates that the models converge to approximately the same forecast (in terms of RMSFE) in each optimization chain. Table 2 presents the absolute values of RMSFE, standard errors (first number in the brackets) and the random initialization variance (second number in the brackets) after this exercise for the models trained with

Table 1: CPI Forecast RMSFE relative to Random Walk

	Test Error	h=2	h=3	h=6	h=12	h=15
AR	0.58 (0.10)	0.69 (0.08)	0.68 (0.09)	0.76 (0.09)	0.87 (0.07)	0.90 (0.07)
NN	0.55 (0.06)	0.65 (0.07)	0.68 (0.05)	0.75 (0.05)	0.91 (0.03)	0.95 (0.02)
LSTM	0.59 (0.08)	0.37 (0.09)	0.28 (0.10)	0.23 (0.08)	0.24 (0.11)	0.27 (0.13)
SARIMA	0.60 (0.13)	0.76 (0.14)	0.80 (0.14)	0.96 (0.18)	1.11 (0.14)	1.11 (0.16)
MS-AR	0.69 (0.12)	1.14 (0.21)	1.17 (0.23)	1.26 (0.26)	1.07 (0.13)	0.73 (0.08)
RW	0.48	0.66	0.87	1.30	2.00	2.19

Note: standard errors are computed by 20 runs of Monte-Carlo cross-validation. The values for RW are absolute and given in %. h indicates the forecast horizon.

ADAM optimizer: simple NN, LSTM and the AR.

Table 2: CPI Forecast RMSFE and Random Initialization Variance

Model	Test Error	h=2	h=3	h=6	h=12
AR	0.28 (0.05, 0.00)	0.46 (0.06, 0.00)	0.59 (0.08, 0.00)	0.98 (0.12, 0.00)	1.74 (0.15, 0.00)
NN	0.27 (0.03, 0.01)	0.43 (0.05, 0.01)	0.60 (0.04, 0.01)	0.98 (0.06, 0.01)	1.82 (0.06, 0.01)
LSTM	0.28 (0.04, 0.04)	0.24 (0.06, 0.04)	0.24 (0.09, 0.04)	0.30 (0.11, 0.07)	0.47 (0.21, 0.17)

Note: Mean and standard deviation of various models after 20 runs of Monte-Carlo cross-validation (first number in the brackets) and after 20 runs on the same data with randomized starting values (second number in the brackets).

AR and NN forecasts have almost no variance due to the initialization which implies that the training algorithm is robust and finds the same optimum in every run. For the LSTM, on contrary, the random-initialization-variance constitutes a large portion of the total variation in the forecast. Two conclusions can be drawn from this observation. First, the loss function of the best performing LSTM is high-dimensional and highly non-linear. It contains 40901[8] parameters and is hard to optimize. The outcome of optimization is rather unstable and depends on the starting points. Second, even withing

[8]The best performing LSTM has 100 hidden units which means that 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. After the last lag of the data 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.

these wide random-initialization bounds the performance of the LSTM is unambiguously better at all forecasting horizons. It might become a caveat if one is interested in a precise parameter identification. However, it does not seem to be a concern since the row model parameters have no meaningful interpretation and can hardly be the ultimate goal of the analysis.

3.2. Real Time Forecasting

After documenting a good performance of the LSTM models 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 2000?". This question is particularly relevant from the policymaker's perspective since the he or she has to make decisions in real-time and therefore need forecasts based on the data available in real-time.

We recursively compute inflation forecasts year by year. Estimations for the year 2001 are based solely on the information up until 2000, including lag length selection, model fitting and cross-validation to select the hyper-parameters and to compute the confidence bounds. Forecasts for $h \geq 2$ are computed by iterating forward the 1-step-ahead forecast.

One-step-ahead rolling window forecasts. In Figure 5 we plot the realization of CPI inflation (red solid line) together with the corresponding forecasts (blue dashed lines). Each point on the forecast line is computed recursively as a one-step-ahead point forecast by training the model on the preceding 480 months (rolling window procedure). The series are aligned such that the vertical distance represents the forecast error. Forecast lines are in general similar but do differ in some periods. Visual inspection suggests that the LSTM forecast is on average more accurate. The statistics in Table 3 confirm more formally that over the entire forecasting period the LSTM's 1-month-ahead forecast is more accurate both in terms of MSFE and MAFE (mean absolute forecast error).

To access the significance of the sample forecast superiority we conduct the Diebold-Mariano test (2002) with modification suggested by Harvey et. al (1997) to account for serial correlation.[9] Given two forecast series the test computes a forecast error loss

[9]The code is taken from John Tsang: https://github.com/johntwk/Diebold-Mariano-Test.

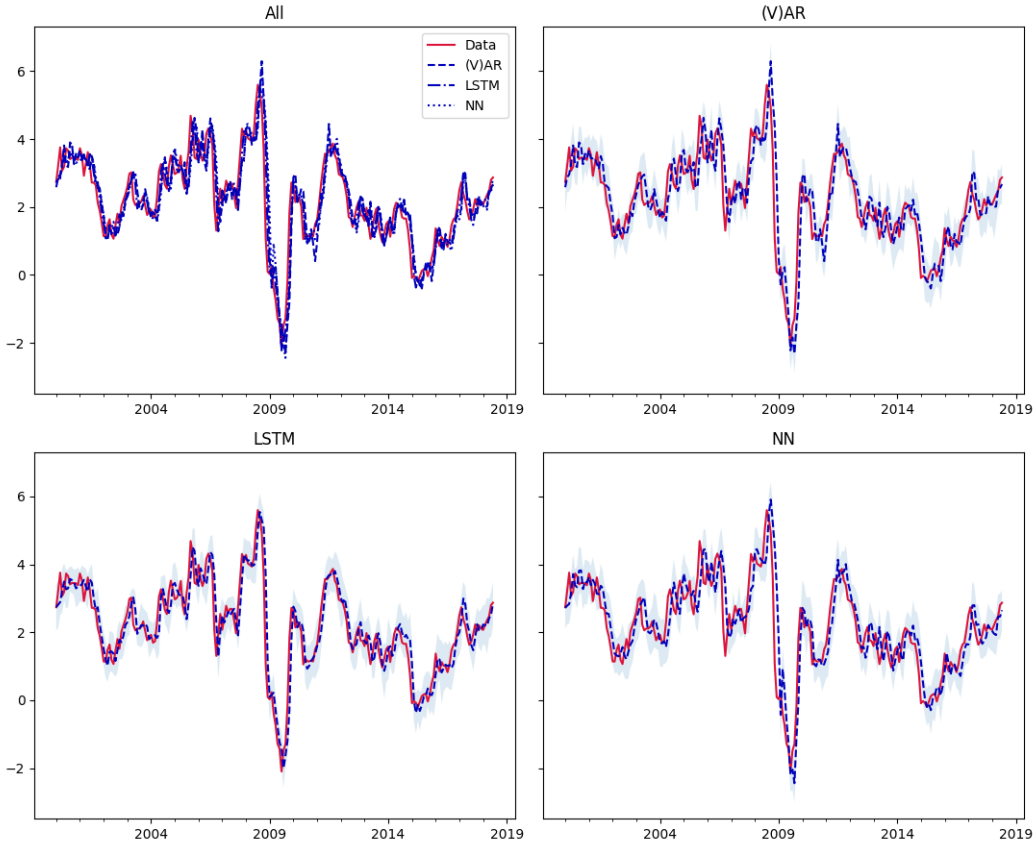

Figure 5: In-Sample Real-Time 1-step-ahead Forecasts, %, 2000:01-2018:12

Table 3: Summary Statistics for Real-Time Forecasts Errors

	MAFE	MSFE	DM test (AR)		DM test (NN)	
			statistics	p-value	statistics	p-value
AR	0.55	0.56	n.a.	n.a.	n.a.	n.a.
LSTM	0.37	0.31	-5.14	0.00	-3.72	0.00
NN	0.46	0.42	-3.61	0.00	n.a.	n.a.

Note: MAFE stands for mean absolute forecast error, MSFE stands for mean squared forecast error. The comparison model for the Diebold-Mariano test is indicated in brackets, MSFE is used as a criterion for the test.

differential $d_{12t} = L(e_{1t}) - L(e_{2t})$ for each t and runs a z-test of the hypothesis that the mean of the differential is zero. The error loss function $L(\cdot)$ is specified by the researcher; we chose quadratic loss, e.i. $L(e_t) = e_t^2$. Under the null, the Diebold-Mariano statistic

is distributed according to a normal distribution:

$$\frac{\bar{d}_{12}}{\hat{\sigma}_{\bar{d}}} \sim N(0,1)$$

Here $\bar{d}_{12}$ is the sample average of the loss differential and the $\hat{\sigma}_{\bar{d}}$ is its sample variance.

Test statistics and p-values are presented in Table 3. The results indicate a significant superiority of the LSTM forecasts. Both mean differentials for the LSTM versus NN and versus AR are negative and significant.

Longer horizons indirect forecasts. Figures 6 through 9 depict several examples of the 1 to 12 month ahead predictions of the different models. Red solid lines depict the real data series and the blue dashed lines are the forecasts. Pictures for all years 2000 through 2018 are presented in the Appendix B.

These pictures confirm the conclusions of Table 1. For some years the gain in accuracy of the LSTM performance is noticeable especially at longer horizons. For example, for years 2002, 2003, 2004, 2006, 2007. However, the accuracy has a high variance. One of the reason for that are the "wild" forecasts that LSTM produces for several years, for example, 2010, 2017. The problem is particularly severe for longer horizons. Implausible forecasts can presumably be eliminated by the researcher through forecast trimming.

4. Sensitivity Analysis

The performance of neural networks depends crucially on the selected hyper-parameters. 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 consider the best performing 10% of each model type and take a closer look at their hyper-parameters.

Table 4, Table 5 and Table 6 present the top 10 performers among AR models, top 10 among NNs and top 10 among LSTMs respectively. Figure 10 and Figure 11 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 hyper-parameter of interest at a time - the number of hidden units, initial learning rate for the optimizer, number of training epochs or the share of the train set - and documented the changes in the model's performance.

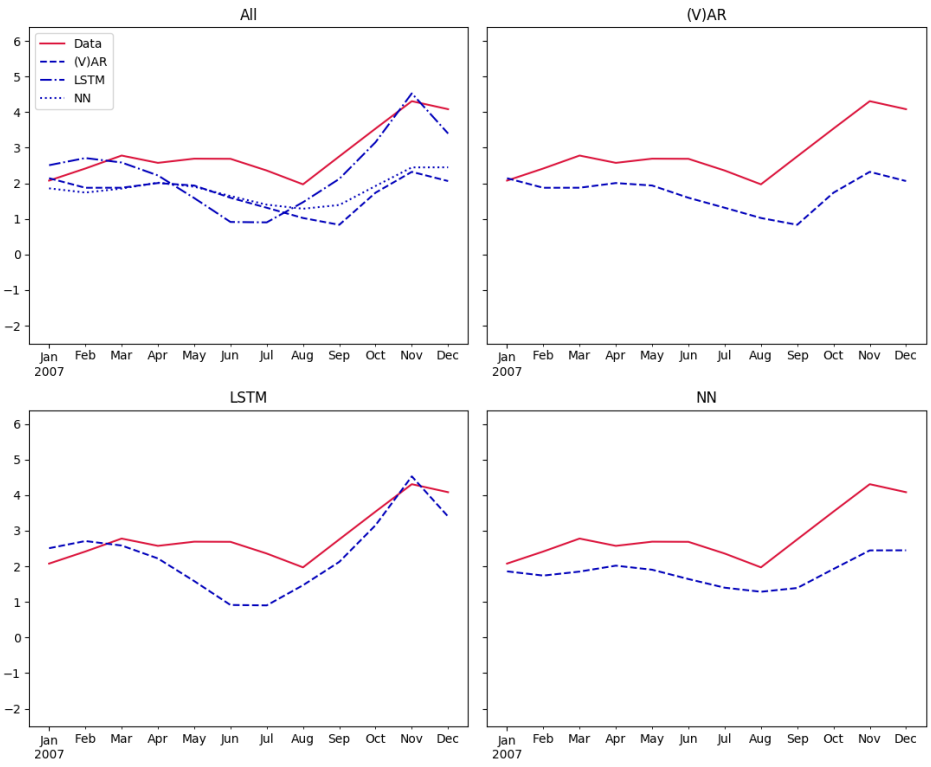

Figure 6: Real-Time Forecast 2007

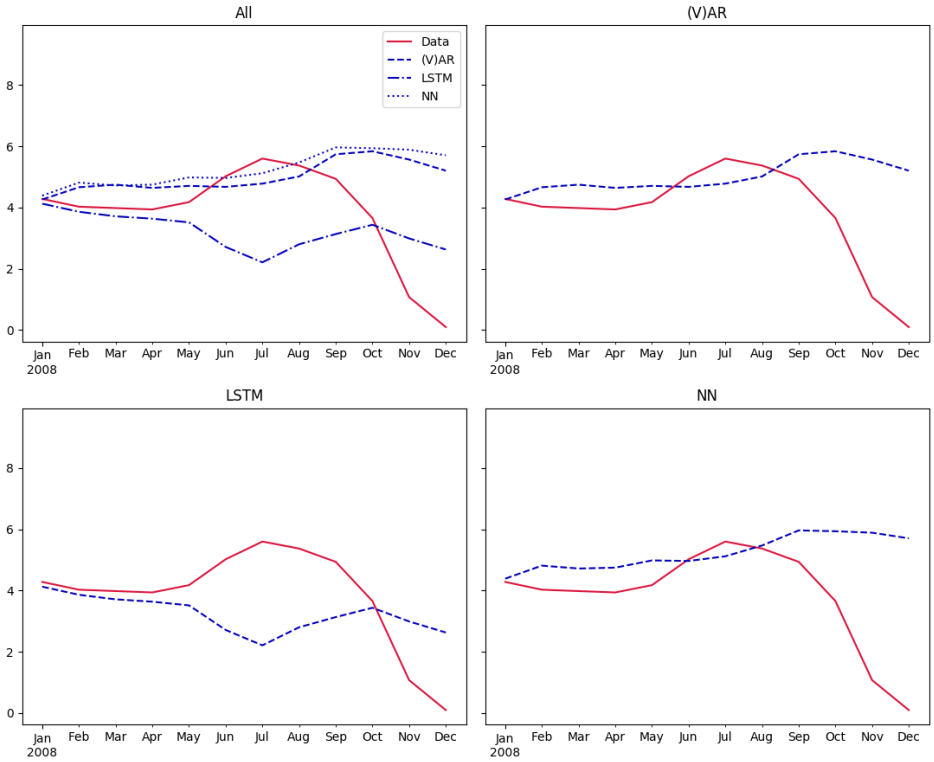

Figure 7: Real-Time Forecast 2008

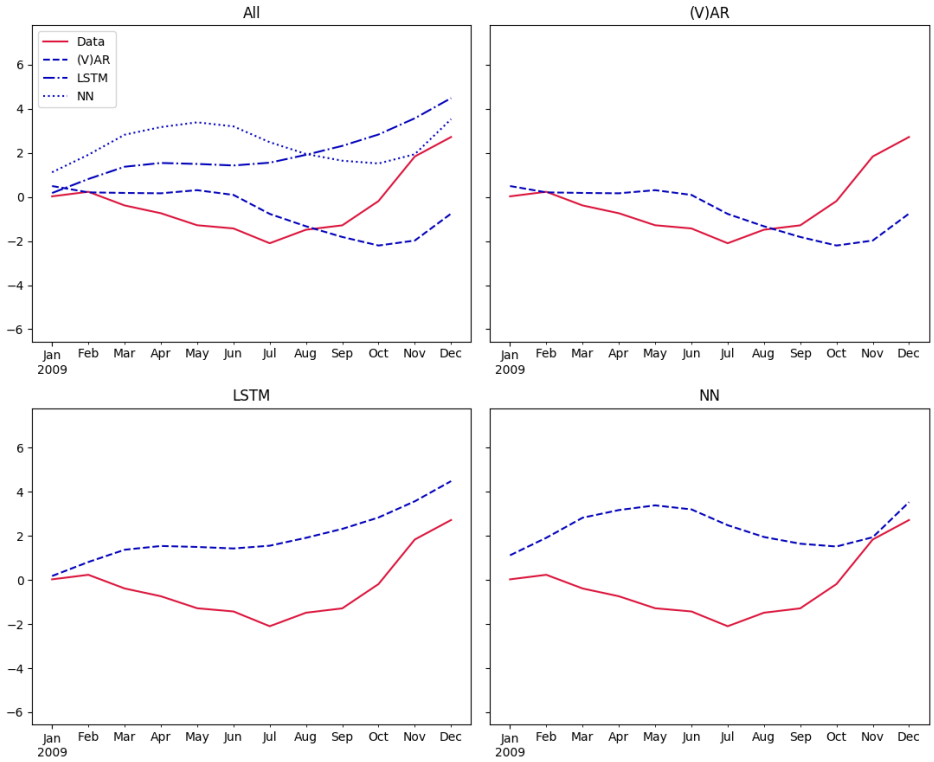

Figure 8: Real-Time Forecast 2009

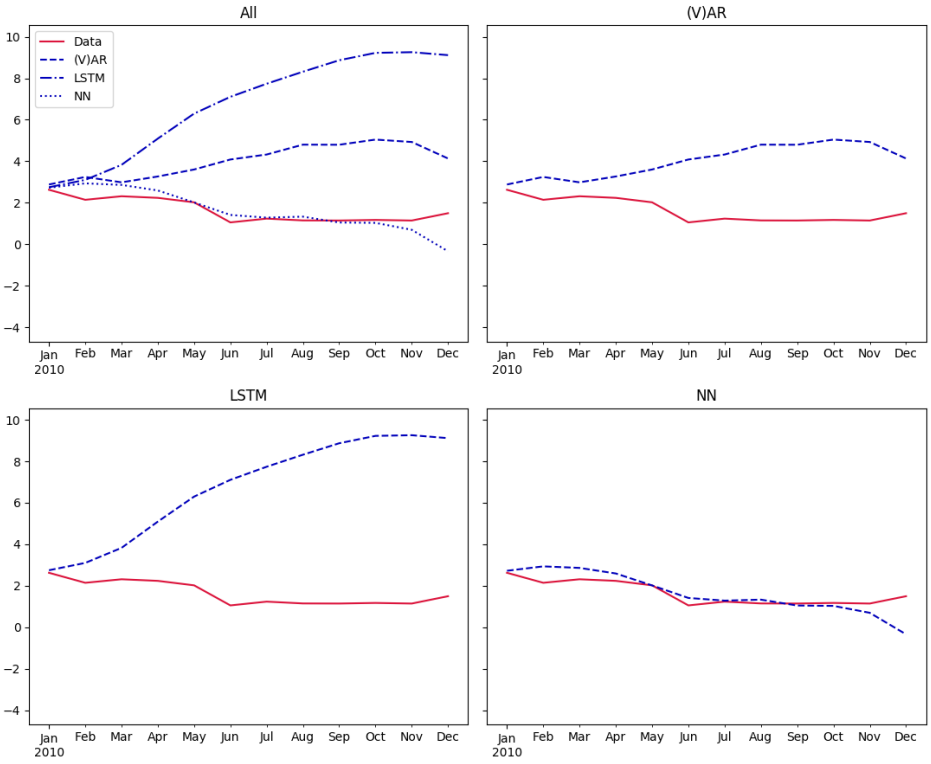

Figure 9: Real-Time Forecast 2010

We now document some recommendations regarding the hyper-parameters for each model type.

AR model. The AR model performs equally well over most lag selection criteria. Based on the results in Table 4 AIC, BIC or fixed number of lags result in virtually the same performance. None of the models with HQIC enters the top 10 list which implies that this does not lead to the good performance. Leaning rate for AR should be at least 0.1 and the network should be trained for at least 5000 epochs.

NN model. As can be seen in Table 5 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 as long as there are more hidden units than lags. Initial learning rate is also irrelevant for the the NN's performance. The bottom left plot on the Figure 10 indicates that the test error of the NN achieves minimum at around 1000 epochs and

Table 4: Top Best Models of AR with Parameters

	Test Error	h=2	h=3	h=6	h=12	h=15	infc	p	Lag	LR	epochs
1	0.28	0.46	0.59	0.98	1.74	1.96	None	24	24	0.3	16000
2	0.29	0.50	0.67	1.01	1.84	2.08	bic	14	24	0.1	16000
3	0.29	0.50	0.66	1.01	1.84	2.08	bic	14	24	0.3	16000
4	0.30	0.42	0.54	0.97	1.79	1.98	None	24	24	0.1	5000
5	0.30	0.45	0.65	1.17	1.70	1.98	aic	19	24	0.1	10000
6	0.30	0.50	0.65	1.00	1.85	2.08	bic	14	24	0.3	10000
7	0.30	0.46	0.64	1.16	1.73	1.98	aic	19	24	0.3	10000
8	0.30	0.46	0.61	1.00	1.77	1.97	None	24	24	0.1	18000
9	0.30	0.49	0.65	0.98	1.84	2.08	bic	14	24	0.1	10000
10	0.30	0.51	0.68	1.02	1.85	2.08	bic	14	24	0.1	18000

Note: selection is based on the average 1-step-ahead RMSFE. 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.

stays unchanged afterwards.[10]

LSTM model. The results for the LSTM are rather mixed with regard to the information criteria. Three best performing models are the ones with a fixed and large number of lags. The maximum number of lags of the top models is 24 which was the largest number we try. The forecast error of LSTM initially decreases with the number of hidden units and then plateaus at around 100. 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 2000 epochs.

In summary, a researcher should bear in mind two aspects when deciding on the LSTM architecture: first the number of hidden units and the maximum number of lags for the information criterion both should not be too small; second the researcher should train the model for a sufficient amount of time. Other hyper-parameters appear not to be much of a concern.

[10]Note that the train error is strictly decreasing in the number of training epochs while the test error is not - the problem called over-fitting. Training of the networks is therefore stopped at the point of the test error minimum and before the train error arrives its minimum. This represents the "early-stopping" principle.

Table 5: Top Best Models of NN with Parameters

	Test Error	h=2	h=3	h=6	h=12	h=15	n	infc	p	Lag	LR	epochs
1	0.27	0.43	0.60	0.98	1.82	2.09	50	bic	14	24	0.00	2000
2	0.27	0.42	0.59	0.96	1.81	2.07	100	bic	14	24	0.00	1000
3	0.27	0.42	0.59	0.97	1.81	2.08	100	bic	14	24	0.00	2000
4	0.27	0.43	0.61	0.96	1.83	2.07	50	bic	14	24	0.00	2000
5	0.27	0.44	0.60	0.99	1.84	2.07	20	bic	14	24	0.00	2000
6	0.27	0.43	0.60	0.98	1.83	2.07	100	bic	14	24	0.00	2000
7	0.28	0.44	0.61	0.99	1.84	2.08	100	bic	14	24	0.00	1000
8	0.28	0.42	0.58	0.94	1.78	2.10	100	bic	14	24	0.01	2000
9	0.28	0.44	0.60	0.95	1.82	2.10	50	bic	14	24	0.00	1000
10	0.28	0.45	0.61	0.95	1.81	2.07	50	bic	14	24	0.01	2000

Note: 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.

Table 6: Top Best Models of LSTM with Parameters

	Test Error	h=2	h=3	h=6	h=12	h=15	n	infc	p	Lag	LR	epochs
1	0.28	0.24	0.24	0.30	0.47	0.60	100	None	24	24	0.03	2000
2	0.28	0.20	0.19	0.23	0.41	0.51	100	None	24	24	0.00	2000
3	0.29	0.24	0.23	0.27	0.43	0.52	100	None	24	24	0.01	2000
4	0.29	0.39	0.45	0.57	1.19	1.48	100	aic	19	24	0.01	1000
5	0.29	0.38	0.46	0.70	1.38	1.67	100	bic	14	24	0.03	2000
6	0.29	0.38	0.45	0.55	1.16	1.42	50	aic	19	24	0.03	2000
7	0.29	0.37	0.44	0.55	1.11	1.41	50	aic	19	24	0.01	2000
8	0.30	0.40	0.47	0.60	1.18	1.48	100	aic	19	24	0.03	1000
9	0.30	0.22	0.23	0.26	0.42	0.55	100	None	24	24	0.10	2000
10	0.30	0.38	0.45	0.52	1.09	1.35	50	aic	19	24	0.00	2000

Note: 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.

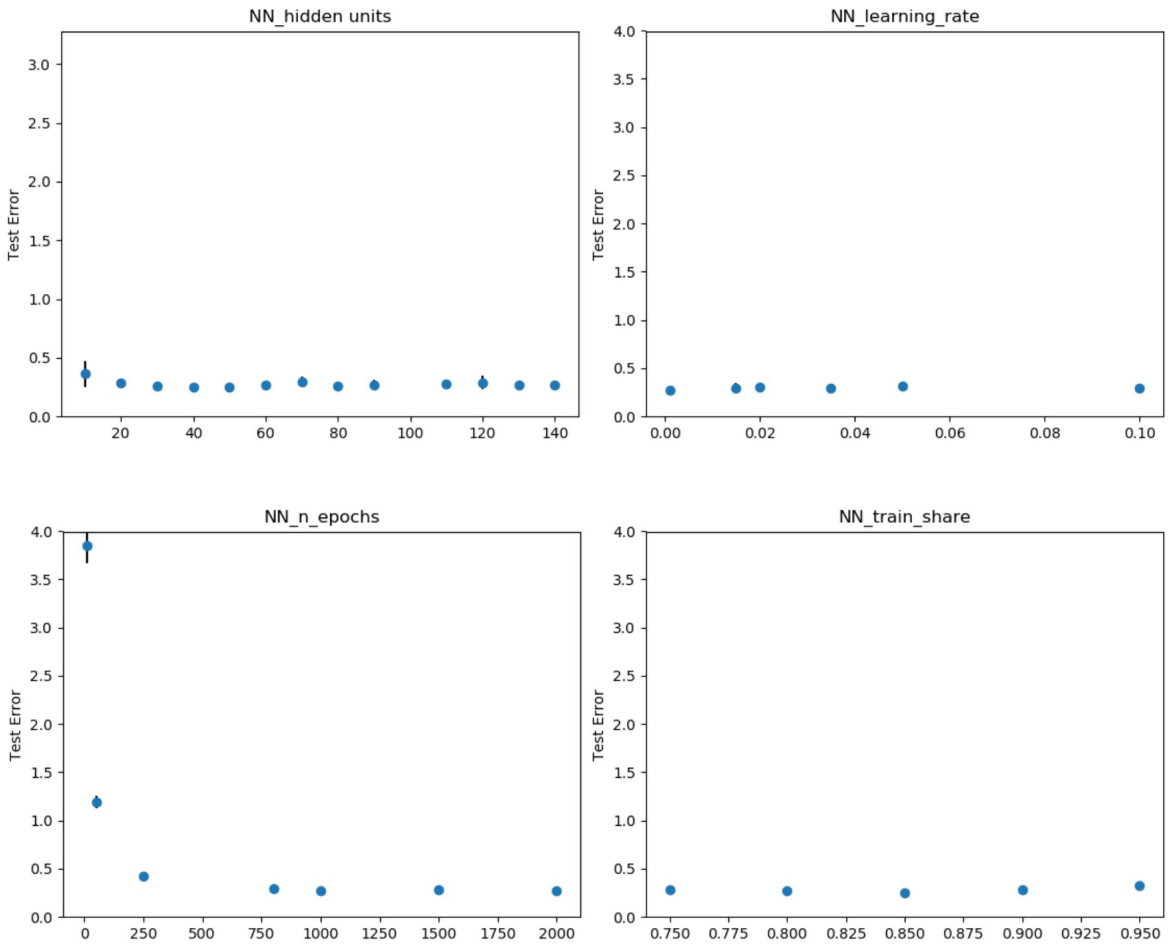

Figure 10: Sensitivity Analysis, Simple NN. Dots indicate mean RMSFE and black bars indicate 90% credible sets after 20 runs of Monte-Carlo cross-validation.

5. 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 outcome. In contrast to linear models it is not possible to directly interpret the weights of the neural networks since the final prediction is a nonlinear function of the network parameters.

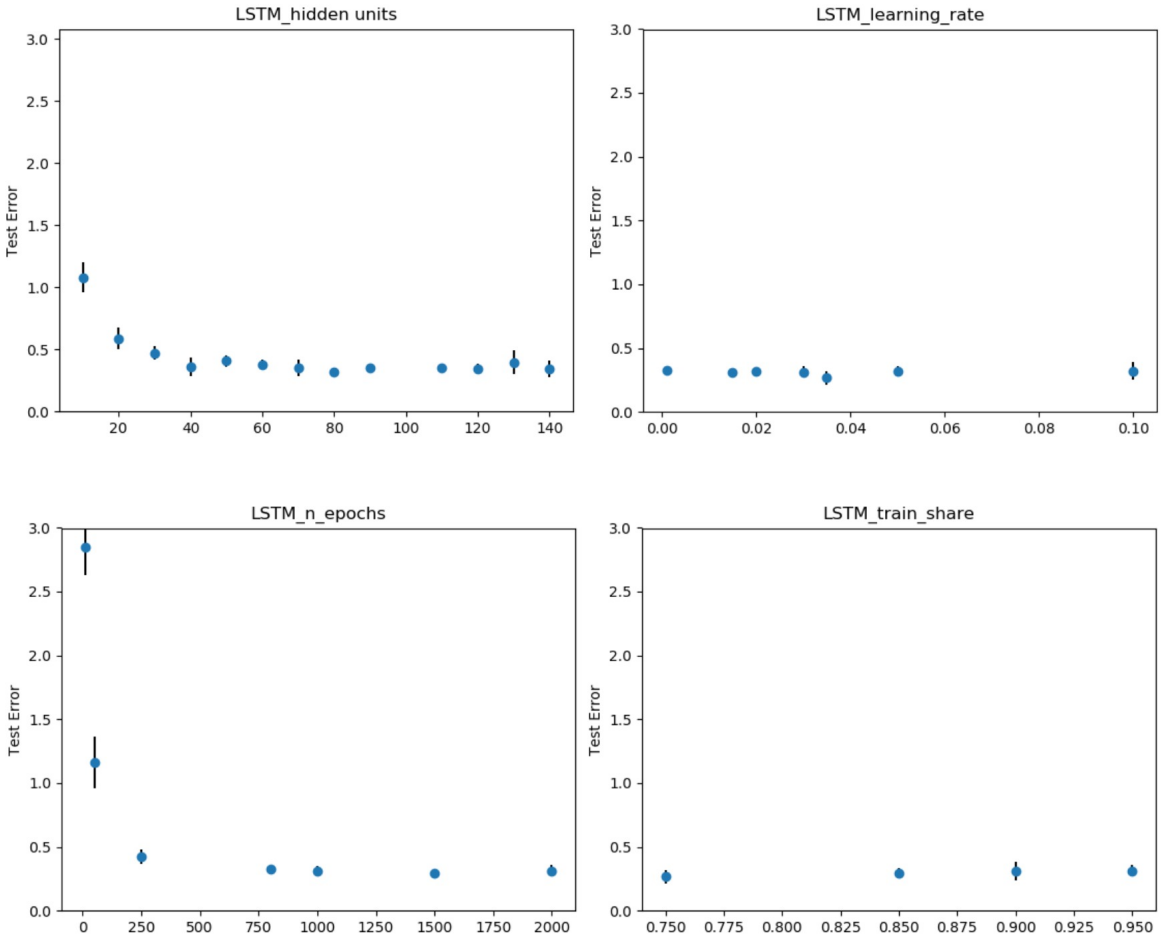

Figure 11: Sensitivity Analysis, LSTM. Dots indicate mean RMSFE and black bars indicate 90% credible sets after 20 runs of Monte-Carlo cross-validation.

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 - lags) 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.

Figure 12 depicts two examples of the LRP measurement for the top performing NN (top row) and the top performing LSTM (bottom row). Additional LRP plots can be found in the Appendix C. 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 14 lags and the NN has 24. The rightmost value of each plot depicted with a dashed line is the 1-step-ahead prediction of the corresponding network. Red bars indicate the lags which 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.

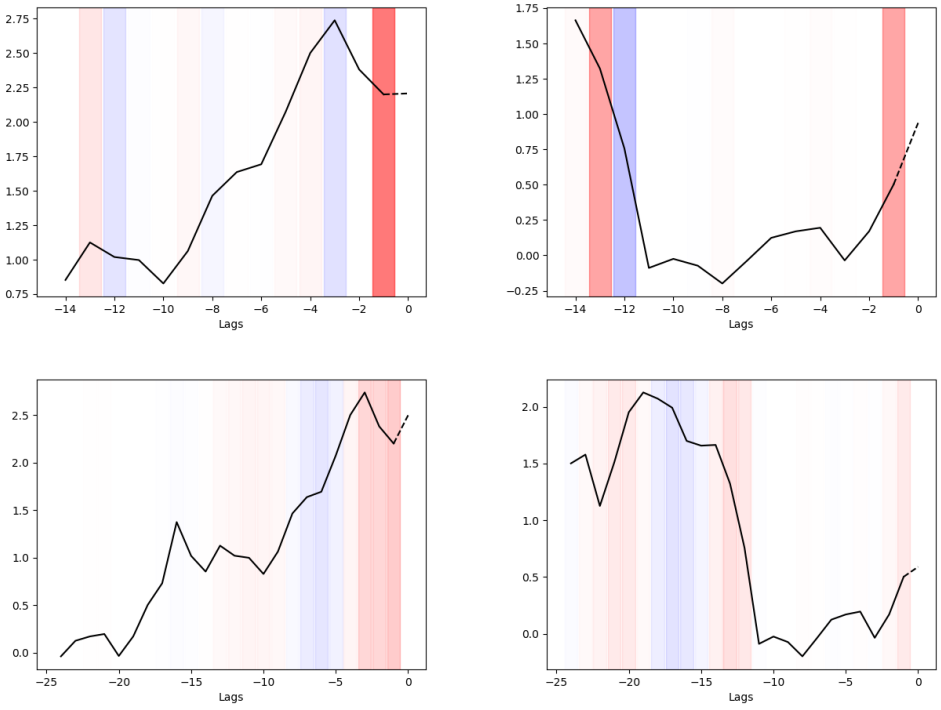

Figure 12: LRP plots for NN (top row) and LSTM (bottom row)

Several observations are worth pointing out. The plots for the NN suggest that the most recent lag (t=-1) as well as the 12th and 13th lags normally contribute the most to the final network outcome. Additionally, the lags with the kinks in the data are sometimes important as, for example, the 3rd lag on the left upper plot. In a nutshell,

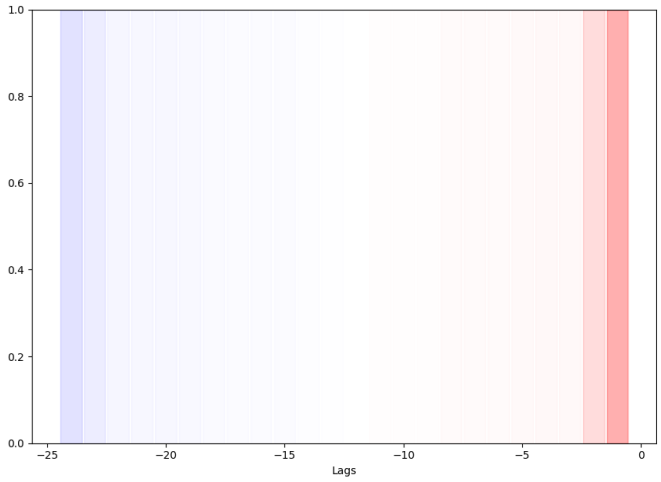

Figure 13: LRP plot for the best AR model

when computing its 1-step-ahead prediction the simple NN takes the average over the most recent lag and the same lag one year ago and then adjusts the forested value if some extreme turns in the trend were observed. It seems like the NN is able to understand the annual nature of the data. It builds a model which is close to an AR(1) in its spirit but adjusted for the annual data structure.

The results for LSTM are more complex. In general the network learns a periodic pattern in which a series of 3-4 blue bars are followed by a series of 3-4 red bars. The most recent lags (1st to 4th) have a positive contribution for the majority of the years, 6th through 8th lags normally contribute negatively and so on. Another observation is that the magnitude of the inflation level at a particular lag seems to be important. Extremely low values often have very small relevance, so the color of the bars for these lags is almost white. While it is not possible to find a simple rule for how the LSTM decides on its forecast value, our LRP analysis suggests that this recurrent neural network fits a nonlinear periodic function and constructs its prediction based on the presence of peaks and declines at particular lags. Additional plots in the appendix also support these conclusions.

The fact that the NN model performs very similar to the regularized AR model (which coefficients are depicted on the Figure 13) signals that it is hard to judge if accounting

for non-linearity automatically improves forecasts (Álvarez-Díaz & Gupta, 2016). The good performance of the LSTM, instead, comes from the combination of non-linearity and a non-sparsity 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.

Moreover, the coefficients for the AR model are fixed for all data points. It means that the contribution of the nth lag to the final prediction is constant or, in other words, this lag always has the same relevance. For the NN model this relevance does not have to be constant but in fact, as we see from the Figure 12, it varies very little from one data point to the other. In contrast, the relevance of the lag n is different for different data points in the LSTM model. The LSTM takes advantage of its flexible architecture and treats different data points differently.

6. Conclusion

This paper evaluates the performance of a nonlinear machine learning method - the long short-term memory recurrent neural network (LSTM) - on the inflation forecasting and compares it to the standard fully connected neural network as well as to the classical methods - the random walk model, linear autoregsression models, seasonal ARIMA model and the Markov switching model.

According to our findings the LSTM outperforms all other forecasting techniques especially at longer horizons with the RMSFE being approximately one third to one halve of the errors for the classical models. We further investigate the real-time forecasting performance of the LSTM by computing 1-month-ahead predictions in a rolling-window setting. We find that its prediction is significantly superior to the NN and VAR forecasts. However, the errors of indirect LSTM forecasts at longer horizons exhibit high variation. LSTMs are extremely accurate in some years and produce "wild forecasts" in others. Implausible forecasts can presumably be eliminated by the researcher through forecast trimming. It 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 show that LSTM models are sensitive to the choice of a few hyper-parameters

and it is therefore worth one's while to tailor these parameters according to the specific forecasting task. Our LRP analysis suggests some intuition for the reasons behind the good performance of the LSTM networks: In addition to capturing the nonlinear features of the data these recurrent neural networks are adjusting to the composition of those inputs that are most relevant for the final prediction. In contrast to the AR and simple NN model LSTMs do not induce sparsity and use all data lags for computing the forecast.

In the light of our results further analysis of the LSTM's performance on predicting other macroeconomic time series as well as inflation prediction in a multivariate-forecasting setting seems to be a promising avenue for future research.

References

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

Álvarez-Díaz, Marcos, & Gupta, Rangan. 2016. Forecasting US consumer price index: does nonlinearity matter? *Applied Economics*, **48**(46), 4462–4475.

Arras, Leila, Montavon, Grégoire, Müller, Klaus-Robert, & Samek, Wojciech. 2017. Explaining recurrent neural network predictions in sentiment analysis. *arXiv preprint arXiv:1706.07206*.

Atkeson, Andrew, & Ohanian, Lee E. 2001. Are Phillips curves useful for forecasting inflation? *Quarterly Review*, 2–11.

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

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

Cybenko, George. 1989. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, **2**(4), 303–314.

Diebold, Francis X, & Mariano, Robert S. 2002. Comparing predictive accuracy. *Journal of Business & economic statistics*, **20**(1), 134–144.

Elger, Thomas, Binner, Jane, Nilsson, Birger, & Tepper, Jonathan. 2006. Predictable non-linearities in U.S. Inflation. **93**(02), 323–328.

Harvey, David, Leybourne, Stephen, & Newbold, Paul. 1997. Testing the equality of prediction mean squared errors. *International Journal of forecasting*, **13**(2), 281–291.

Hochreiter, Sepp, & Schmidhuber, Jürgen. 1997. Long short-term memory. *Neural computation*, **9**(8), 1735–1780.

Kingma, Diederik P., & Ba, Jimmy. 2014. Adam: A Method for Stochastic Optimization. *CoRR*, **abs/1412.6980**.

Kock, Anders Bredahl, Teräsvirta, Timo, *et al.* 2011. *Forecasting macroeconomic variables using neural network models and three automated model selection techniques*. Tech. rept. Department of Economics and Business Economics, Aarhus University.

Kuan, Chung-Ming, & Liu, Tung. 1995. Forecasting exchange rates using feedforward and recurrent neural networks. *Journal of applied econometrics*, **10**(4), 347–364.

Lapuschkin, Sebastian, Binder, Alexander, Montavon, Grégoire, Müller, Klaus-Robert, & Samek, Wojciech. 2016. The LRP toolbox for artificial neural networks. *The Journal of Machine Learning Research*, **17**(1), 3938–3942.

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

Makridakis, Spyros, Spiliotis, Evangelos, & Assimakopoulos, Vassilios. 2018. Statistical and Machine Learning forecasting methods: Concerns and ways forward. *PloS one*, **13**(3), e0194889.

Marcellino, Massimiliano, Stock, James H, & Watson, Mark 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, Peter, & McNelis, Paul. 2005. Forecasting inflation with thick models and neural networks. *Economic Modelling*, **22**(5), 848–867.

Medeiros, Marcelo C, Vasconcelos, Gabriel, Veiga, Alvari, & Zilberman, Eduardo. 2018. Forecasting Inflation in a Data-Rich Environment: The Benefits of Machine Learning Methods. *SSRN working paper*.

Nair, Vinod, & Hinton, Geoffrey E. 2010. Rectified linear units improve restricted boltzmann machines. *Pages 807–814 of: Proceedings of the 27th international conference on machine learning (ICML-10)*.

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

Stock, James H, & Watson, Mark W. 1998 (June). *A Comparison of Linear and Nonlinear Univariate Models for Forecasting Macroeconomic Time Series*. Working Paper 6607. National Bureau of Economic Research.

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

Stock, James H, & Watson, Mark W. 2007. Why has US inflation become harder to forecast? *Journal of Money, Credit and banking*, **39**, 3–33.

Swanson, Norman R, & White, Halbert. 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, Norman R, & White, Halbert. 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, Timo. 2006. Forecasting economic variables with nonlinear models. *Handbook of economic forecasting*, **1**, 413–457.

Teräsvirta, Timo, & CASE, Humboldt. 2017. Nonlinear models in macroeconometrics. *In: Oxford Research Encyclopedia in Economics and Finance*. Oxford University Press.

Appendices

A. CPI Inflation Data

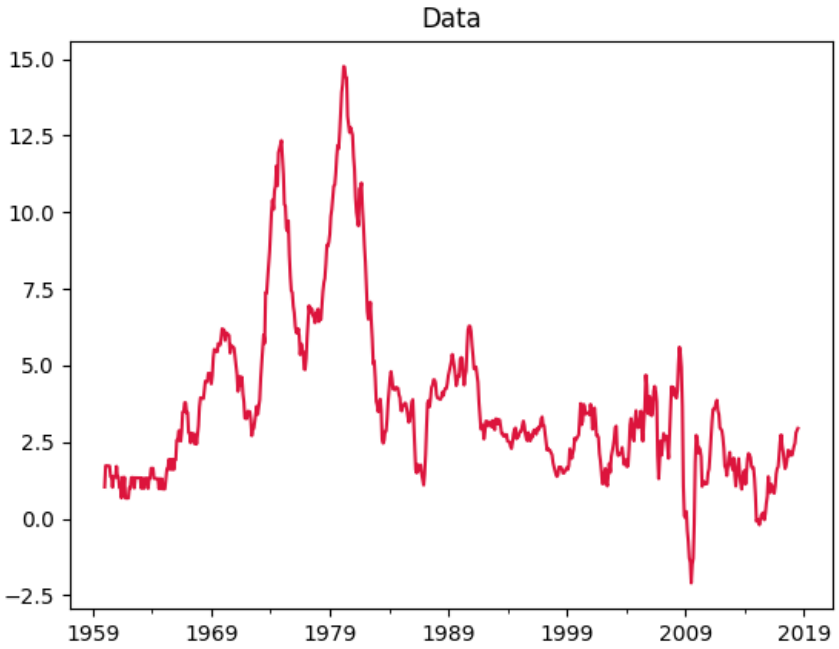

Figure 14: Annual CPI Inflation, % , Monthly frequency, from 1960-01 to 2018-07

Augmented Dicky-Fuller Test:

ADF-statistics:	-3.06
p-value:	0.02

B. 12-Months-Ahead Forecasts

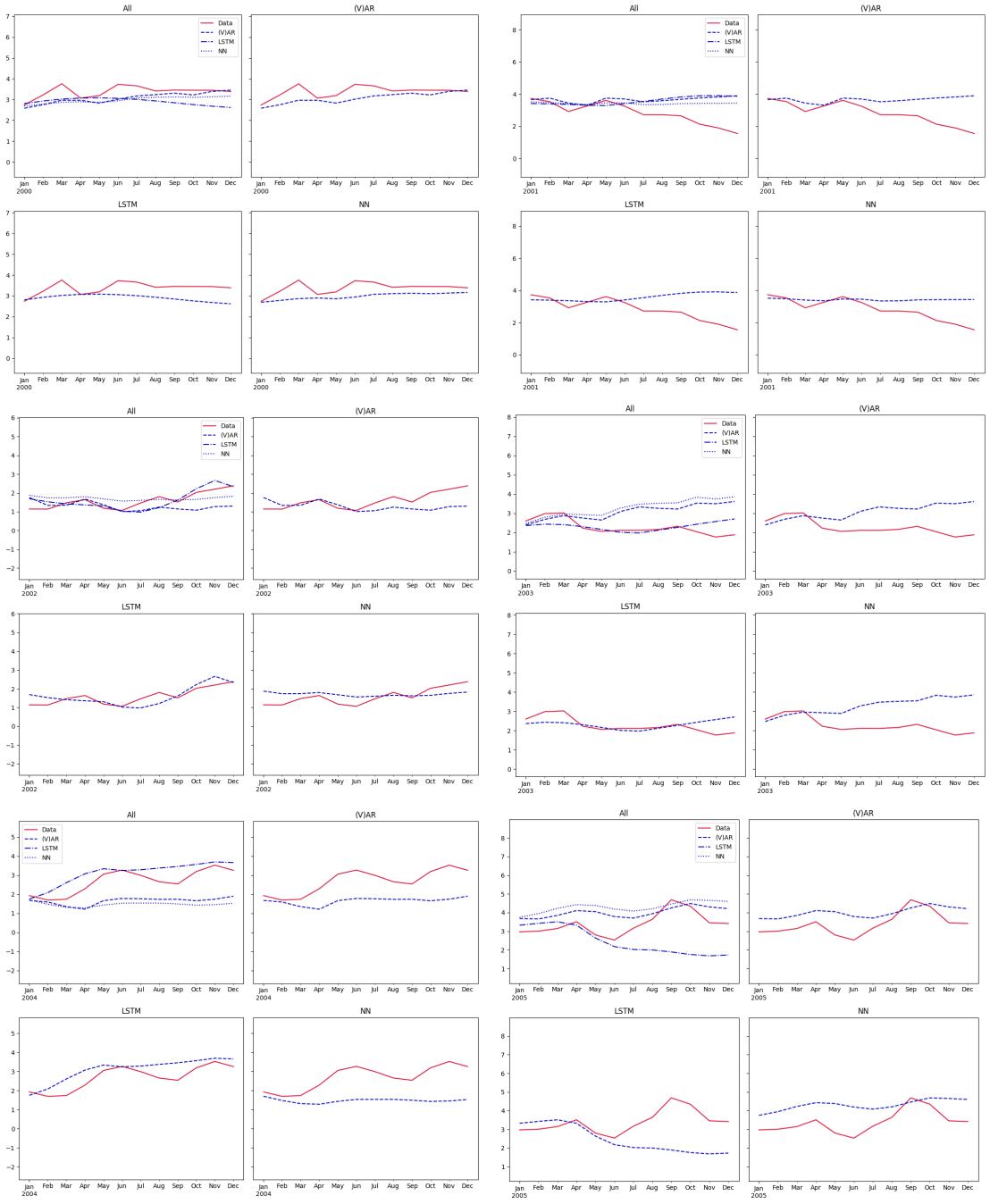

Figure 15: Real Time Forecasts, Forecast Step h=1:12, 2000-2005

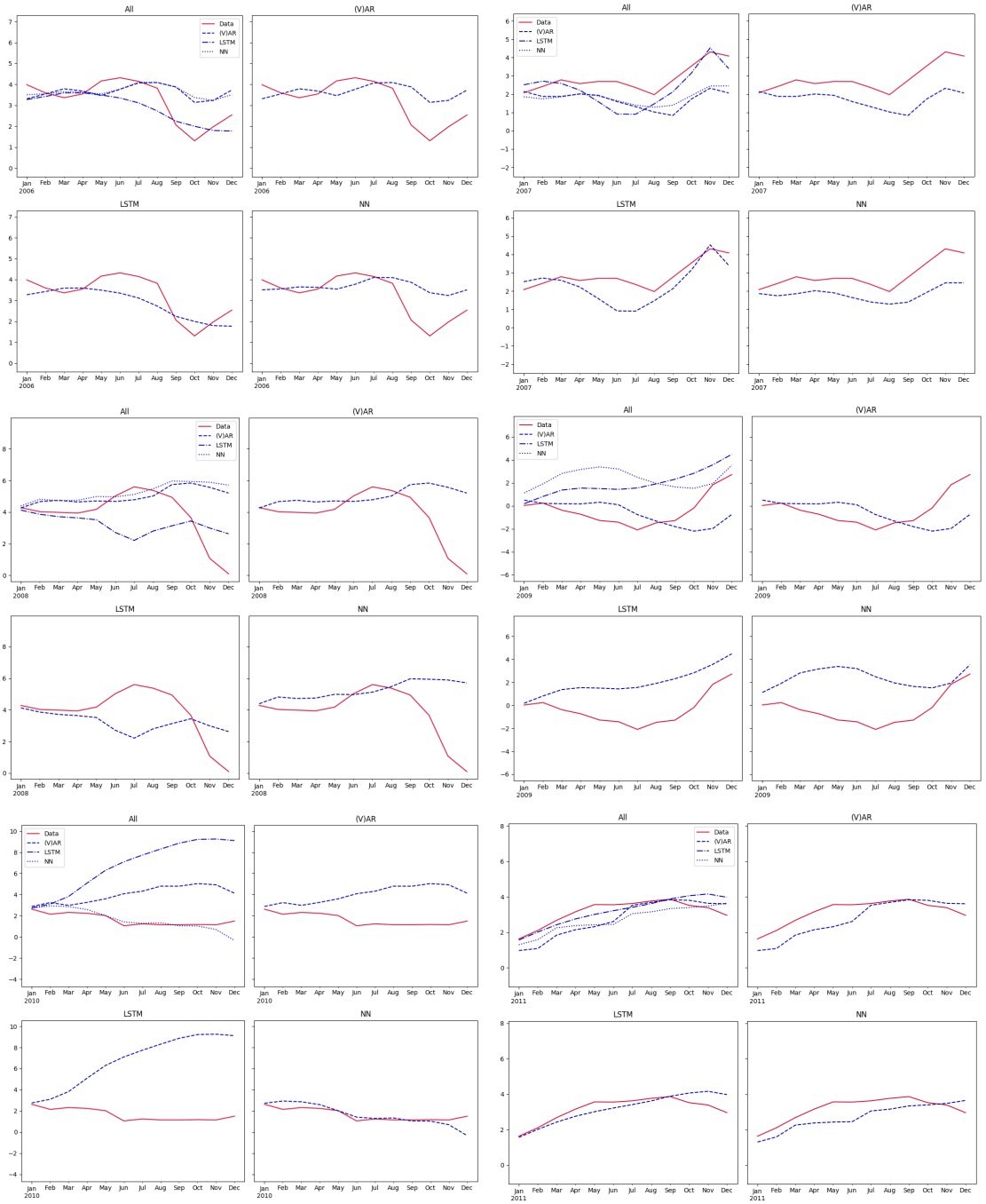

Figure 16: Real Time Forecasts, Forecast Step h=1:12, 2006-2011

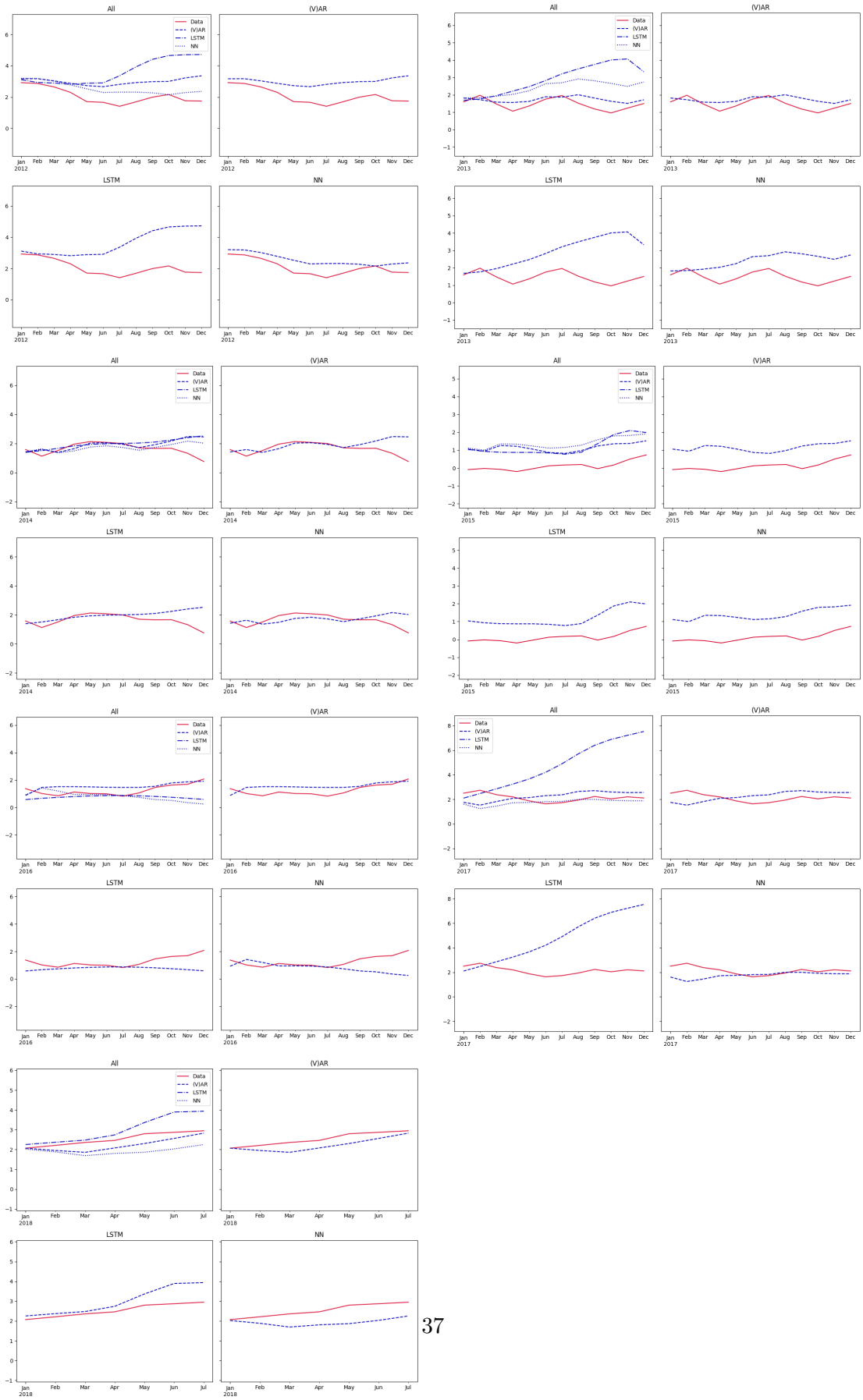

Figure 17: Real Time Forecasts, Forecast Step h=1:12, 2012-2018:07

C. LRP Analysis

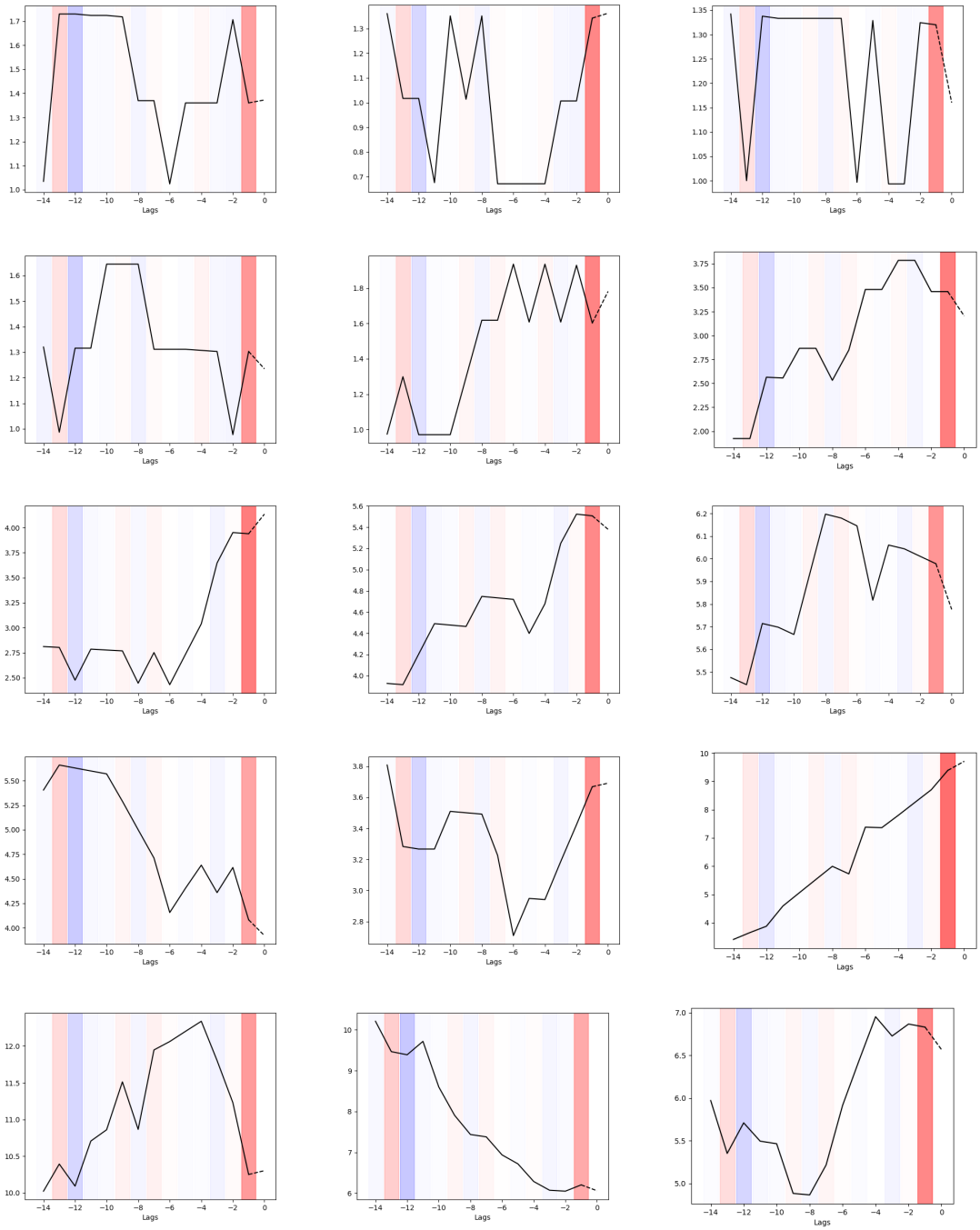

Figure 18: Layer-wise Relevance Propagation Analysis for Simple NN, Examples (part 1)

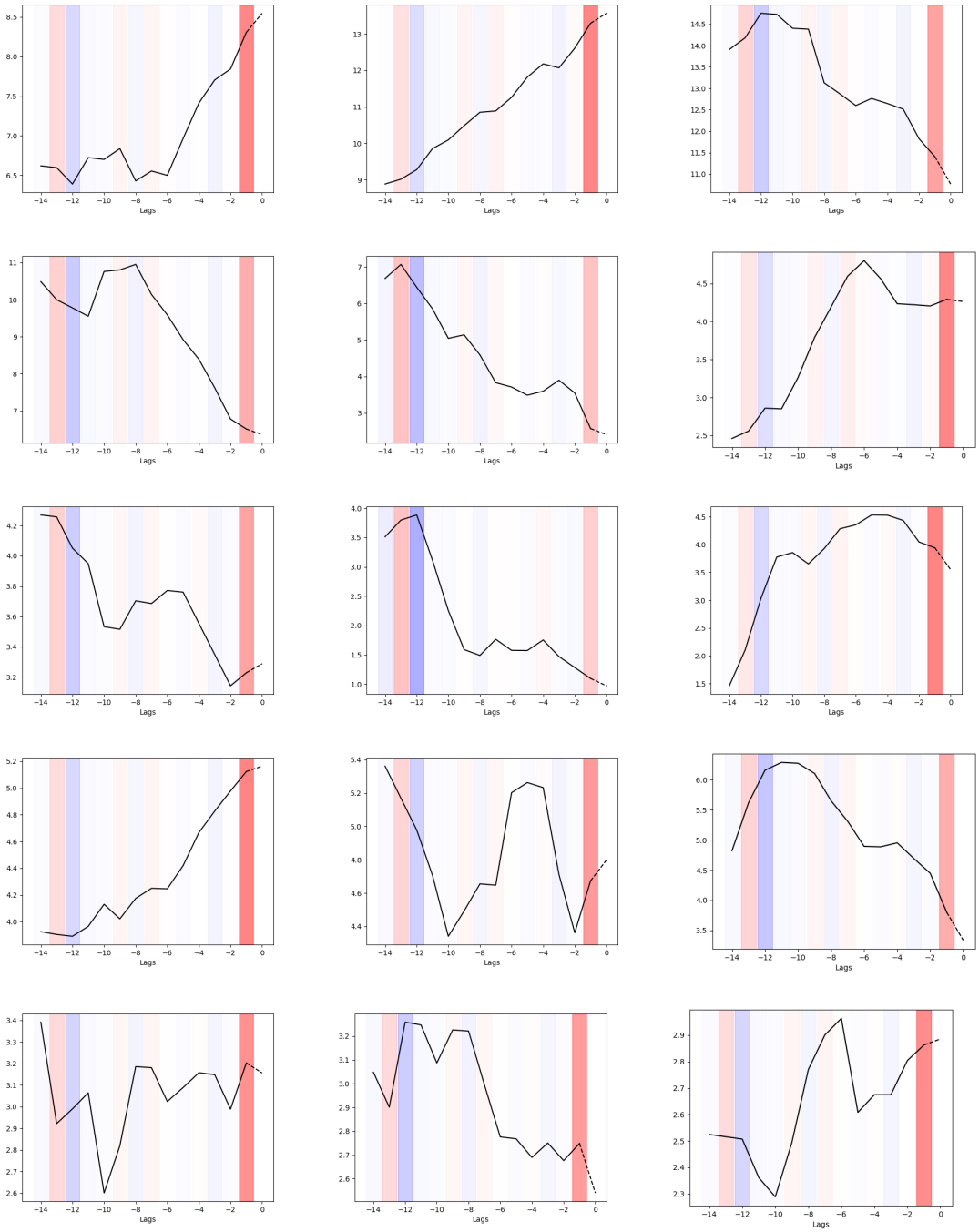

Figure 19: Layer-wise Relevance Propagation Analysis for Simple NN, Examples (part 2)

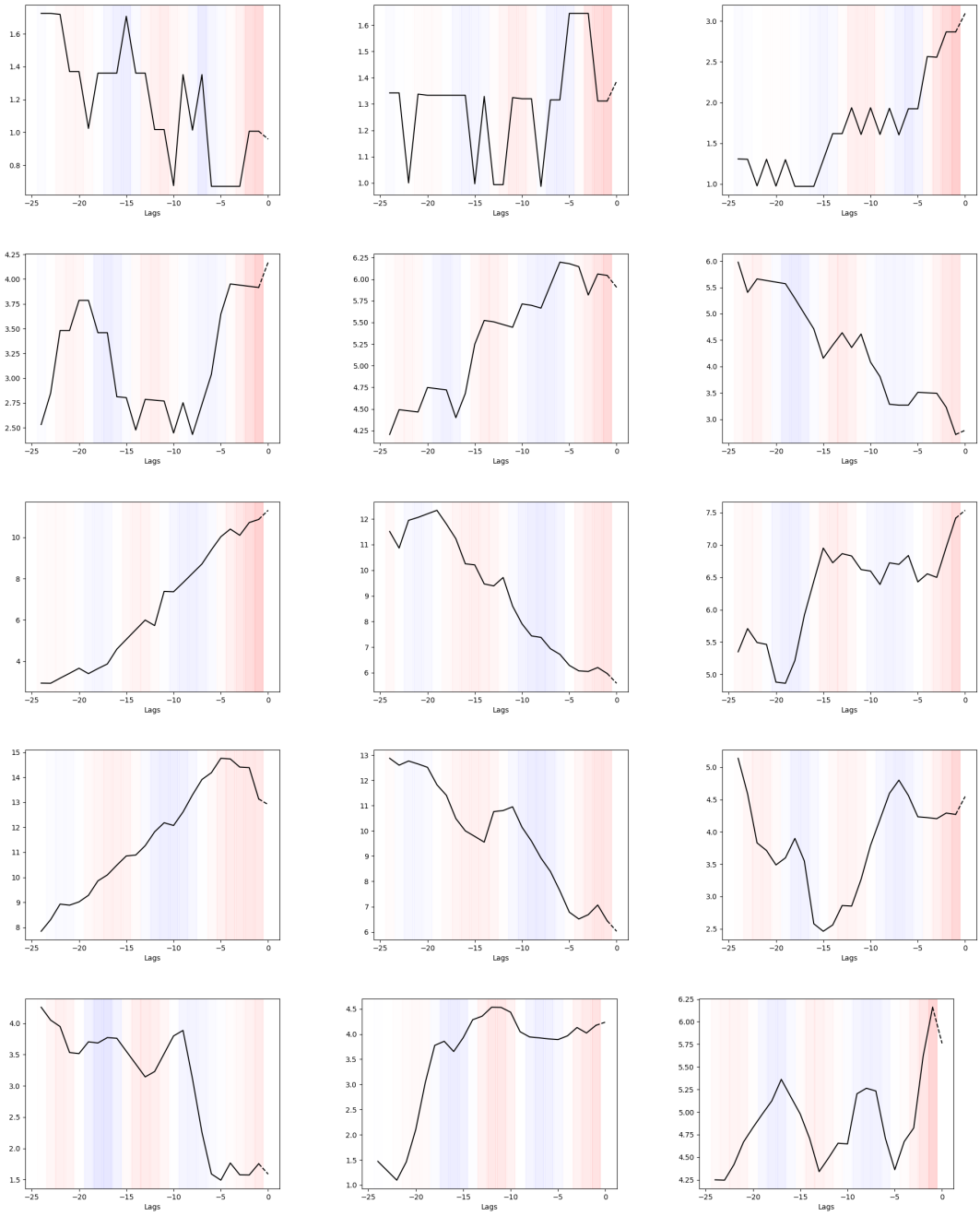

Figure 20: Layer-wise Relevance Propagation Analysis for LSTM, Examples (part 1)

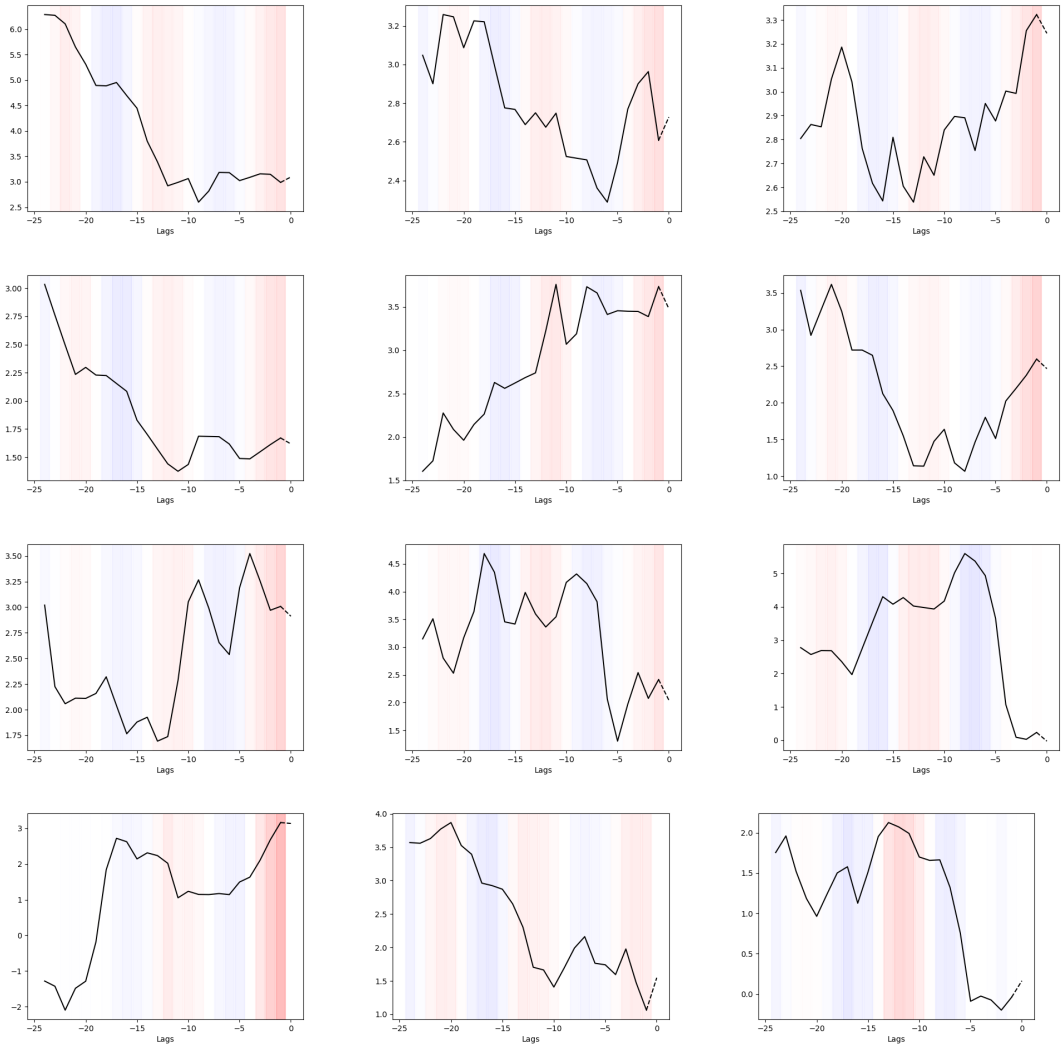

Figure 21: Layer-wise Relevance Propagation Analysis for LSTM, Examples (part 2)

D. Results for Personal Consumption Expenditures (PCE)

In this section we present the results for the data on PCE inflation - presumably the primary inflation indicator that the US central bank uses to develop economic policy. These results were created in the same way as the ones presented above. The general conclusion of the main text remains the same.

As can be seen in Table 7 and Table 8, all models systematically perform better than random walk, The LSTM model shows the best performance. On this dataset the LSTM is not out-performing the simpler NN and the AR model significantly indicating that the more sophisticated structure of the LSTM is not necessary for this data.

Table 7: PCE Forecast RMSFE relative to Random Walk

	Test Error	h=2	h=3	h=6	h=12
AR	0.25 (0.05)	0.32 (0.05)	0.38 (0.04)	0.48 (0.07)	0.43 (0.08)
LSTM	0.28 (0.04)	0.29 (0.04)	0.29 (0.04)	0.30 (0.05)	0.34 (0.07)
NN	0.25 (0.05)	0.29 (0.04)	0.34 (0.05)	0.47 (0.07)	0.41 (0.08)
SARIMA	0.24 (0.06)	0.29 (0.07)	0.30 (0.07)	0.35 (0.08)	0.51 (0.10)
MS-AR	0.37 (0.10)	0.40 (0.11)	0.39 (0.11)	0.62 (0.35)	2.47 (0.79)
RW	0.48	0.66	0.87	1.30	2.00

Note: standard errors are computed by 20 runs of Monte-Carlo cross-validation. The values for RW are absolute and given in %. h indicates the forecast horizon.

Table 8: PCE Forecast RMSFE and Random Initialization Variance

Model	Test Error	h=2	h=3	h=6	h=12
AR	0.12 (0.02, 0.00)	0.21 (0.03, 0.00)	0.34 (0.04, 0.00)	0.62 (0.09, 0.00)	0.87 (0.17, 0.00)
LSTM	0.14 (0.02, 0.06)	0.19 (0.03, 0.05)	0.26 (0.03, 0.05)	0.39 (0.06, 0.05)	0.67 (0.14, 0.15)
NN	0.12 (0.02, 0.00)	0.19 (0.03, 0.01)	0.30 (0.04, 0.02)	0.60 (0.09, 0.02)	0.82 (0.15, 0.01)

Note: Mean and standard deviation of various models after 20 runs of Monte-Carlo cross-validation (first number in the brackets) and after 20 runs on the same data with randomized starting values (second number in the brackets).

D.1. Real Time Forecasting

Figure 22 shows the results of the real time forecast exercise. From Table 9 one can see that the methods are very close to each other in terms of performance. This is in accordance to the results in Table 7, since the real time forecasting is a one-step ahead exercise. On this forecast horizon the crossvalidation also showed only small differences between the models.

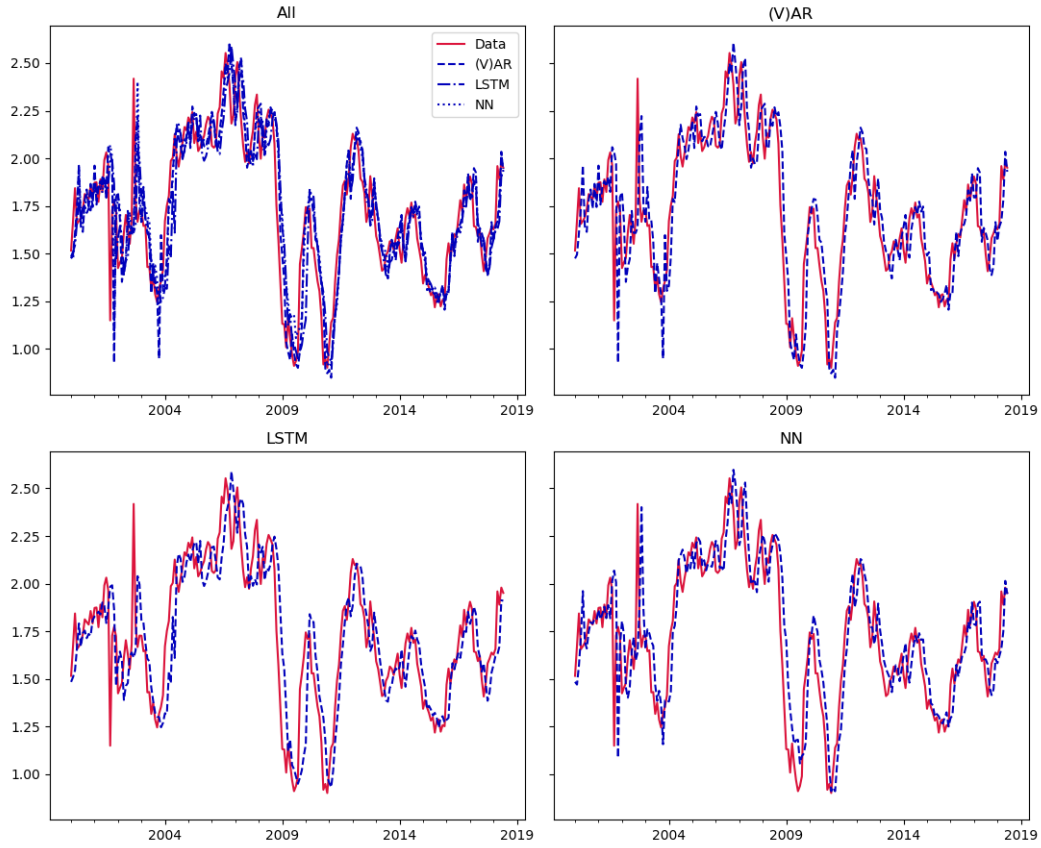

Figure 22: In-Sample Real-Time 1-step-ahead Forecasts, %, 2000:01-2018:12

D.2. Sensitivity Analysis

The best performing combinations of hyperparameters can be seen in Tables 10, 11 and 12. The models showed the same patterns in the sensitivity of the performance to certain parameter changes as before with the PCI data.

Table 9: Summary Statistics for Real-Time Forecasts Errors

	MAFE	MSFE	DM test (AR)		DM test (NN)	
			statistics	p-value	statistics	p-value
AR	0.13	0.03	-1.36	0.18	-0.77	0.45
LSTM	0.15	0.04	n.a.	n.a.	1.12	0.26
NN	0.14	0.04	n.a.	n.a.	n.a.	n.a.

Note: MAFE stands for mean absolute forecast error, MSFE stands for mean squared forecast error. The comparison model for the Diebold-Mariano test is indicated in brackets, MSFE is used as a criterion for the test.

Table 10: Top Best Models of AR with Parameters

	Test Error	h=2	h=3	h=6	h=12	h=15	infc	p	Lag	LR	epochs
1	0.12	0.21	0.34	0.62	0.87	0.80	None	24	24	0.30	18000
2	0.12	0.21	0.33	0.61	0.84	0.80	None	24	24	0.05	18000
3	0.13	0.24	0.36	0.61	0.87	0.95	None	19	19	0.10	16000
4	0.13	0.23	0.35	0.65	1.14	1.21	hqic	14	19	0.10	16000
5	0.13	0.24	0.35	0.63	0.89	0.94	None	19	19	0.10	18000
6	0.14	0.22	0.34	0.63	0.85	0.78	None	24	24	0.10	18000
7	0.14	0.24	0.35	0.65	1.15	1.22	hqic	14	14	0.30	10000
8	0.14	0.26	0.36	0.64	0.88	0.93	None	19	19	0.30	18000
9	0.14	0.23	0.34	0.63	0.85	0.80	None	24	24	0.30	10000
10	0.14	0.23	0.34	0.62	0.82	0.78	None	24	24	0.05	10000

Note: selection is based on the average 1-step-ahead RMSFE. 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 11: Top Best Models of NN with Parameters

	Test Error	h=2	h=3	h=6	h=12	h=15	n	infc	p	Lag	LR	epochs
1	0.12	0.19	0.30	0.60	0.82	0.75	50	None	24	24	0.03	3000
2	0.13	0.21	0.28	0.55	1.05	1.22	100	aic	14	24	0.01	2000
3	0.13	0.21	0.31	0.59	1.09	1.18	20	aic	14	24	0.03	3000
4	0.13	0.21	0.30	0.57	0.98	1.08	100	bic	14	24	0.00	3000
5	0.13	0.20	0.29	0.56	1.02	1.13	100	aic	14	24	0.03	3000
6	0.13	0.19	0.27	0.53	0.98	1.15	100	bic	14	24	0.01	2000
7	0.13	0.20	0.29	0.58	1.03	1.16	50	aic	14	24	0.01	2000
8	0.13	0.22	0.32	0.60	1.07	1.21	50	hqic	5	12	0.03	3000
9	0.13	0.22	0.32	0.61	1.07	1.15	100	bic	14	24	0.00	2000
10	0.13	0.21	0.32	0.61	1.07	1.19	20	aic	14	24	0.00	3000

Note: 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.

Table 12: Top Best Models of LSTM with Parameters

	Test Error	h=2	h=3	h=6	h=12	h=15	n	infc	p	Lag	LR	epochs
1	0.14	0.19	0.26	0.39	0.67	0.81	100	None	12	12	0.01	3000
2	0.14	0.18	0.20	0.30	0.48	0.55	100	None	24	24	0.03	3000
3	0.14	0.19	0.24	0.37	0.66	0.81	100	hqic	14	24	0.03	2000
4	0.14	0.19	0.24	0.38	0.67	0.80	20	None	24	24	0.03	3000
5	0.14	0.22	0.29	0.44	0.79	0.94	20	aic	7	12	0.03	3000
6	0.14	0.19	0.24	0.37	0.64	0.80	100	bic	14	24	0.01	3000
7	0.14	0.23	0.30	0.49	0.89	1.04	20	bic	5	12	0.01	3000
8	0.14	0.19	0.24	0.36	0.66	0.79	50	bic	14	24	0.01	3000
9	0.14	0.19	0.24	0.39	0.73	0.90	100	bic	14	24	0.03	3000
10	0.15	0.22	0.29	0.44	0.77	0.89	50	bic	5	12	0.00	3000

Note: 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.