Forecasting Inflation with Recurrent Neural Networks

(PRELIMINARY)

Anna Almosova*, Niek Andresen†

October 2018

This paper demonstrates that machine learning techniques can be used to efficiently forecast macroeconomic time series. We show that artificial neural networks outperform a linear autoregressive (AR) and a random walk (RW) models in forecasting the monthly US CPI inflation. One-step-ahead RMSE of a simple neural network (NN) and of a long short-term memory (LSTM) recurrent neural network is approximately half of the corresponding measure for the AR or RW models. For short horizons, up to 3 steps ahead, both NN and LSTM give more accurate forecasts than the AR and RW. At longer horizons, 6 through 12 steps ahead, the performance of the NN becomes on a par with the linear AR and RW models. However, the LSTM continues to produce more accurate predictions with the errors being approximately 40% smaller than of the RW or AR. 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.

Keywords: Recurrent 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

1 Introduction

Accurate inflation forecasting is key for many economic decisions. Private investors need to know future inflation to adjust their asset holdings, firms need to forecast the aggregate inflation level to adjust their prices and maximize profits, central banks need to forecast inflation to conduct an efficient monetary policy.

Inflation forecasting is an interesting yet challenging task. As Stock & Watson (2007) point out, inflation has recently become both easier and harder to predict. On the one hand, since the mid-80s inflation in the US 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 a nonlinear nonparametric method from the Machine Learning literature, that is novel in this context - a long short-term memory recurrent neural network (LSTM RNN). We see four main advantages of this method. First, RNNs are flexible and data-driven. It means that the researcher does not have to specify the exact form of the nonlinearity. Instead the RNN will infer it from the data itself. Second, as stated by the universal approximation theorem (Cybenko (1989)), under some mild regularity conditions RNNs 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, RNNs 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 RNNs to predict the last word in the sentence and help the user while she is typing a message.

[1] For example Stock & Watson (2007) fit an integrated moving average (time-varying trend-cycle) model to the GDP inflation 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".

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 RNN with a simple fully-connected NN model, an AR(p) model and a RW-type model on monthly inflation forecasting on the US data.[3] 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. 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, both the NN and the RNN outperform the linear forecast from the AR and RW. At all forecast horizons AR models perform very similar to the RW. At shorter horizons, such as 1 - 3 step-ahead forecasts on monthly data, the root mean squared forecast error (RMSFE) of the AR and RW models is almost double as high as the RMSFE of the neural networks. At longer horizons, 6 through 12 steps ahead, the performance of the simple NN becomes on a par with the linear RW model. However, the RNN continues to make more accurate predictions with the error being approximately 40% smaller then of the RW. Our findings thus suggest that the LSTM RNN is an efficient nonlinear model for forecasting inflation especially at longer horizons.

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

[2] We use TensorFlow on Python.

[3] As a part of our contribution we made a toolkit for NN and RNN forecasting available on our github repository.

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 results for the RNN are rather mixed with regard to the information criteria and the maximum number of lags. The best performing model is the one with a fixed and large number of lags. The accuracy of the RNN 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.

Overall, the outcome of the simple NN is more robust whereas the RMSFE of the RNN can change significantly with different hyper-parameters. We conclude that it is worth one's while to spend time on the fine-tuning of the RNN model according to the problem at hand. It is also desirable to use a large amount of hidden units in the LSTM RNN. Of course, it comes at the cost of computation time required to train this network.

Our LRP analysis indicates that both NNs and RNNs mostly pay attention to the most recent lag and the same lag one year ago when computing their predictions. It seems like neural networks are able to detect the annual seasonality in the data and take this into account. Additionally, on average the periods with sharp kinks in the time series contribute a lot to the final prediction. These periods inform the network whether the current trend in the data is positive or negative.

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 NNs and RNNs for the exchange rate forecasting in several countries, Swanson & White (1997b) and Swanson & White (1997a) 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 based on various performance measures. Elger *et al.* (2006) shows that for shorter horizons RNNs are comparable with Markov switching autoregressive models and at longer horizons Markov switching models are more accurate. Similarly to our analysis they study an RNN for inflation prediction. 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.

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. We also use a different optimization algorithm to fit the NN and RNN models - the Adam optimizer. Our choice of the optimization routine is based on its success in machine learning applications. Moreover, most 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 RNN. 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.

2 Methodology

We rank the forecasting methods based on the RMSFE for out-of-sample forecast on the test set. The share of the test set is 10% of the whole data sample. Test sets are constructed by randomly drawing non overlapping data sequences of the length p, where p is the selected lag length. The same procedure is applied to obtain validation sets for the sensitivity analysis and real-time forecasts (Monte Carlo cross validation). Note that since our test data samples are 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.

We focus on the 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 country-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.

Our model can be summarized as:

$$
\begin{aligned}
&y_t = f\left(Z_{t-1}, W\right) + u_t, \text{where } Z_{t-1} = [y_{t-1}, ...y_{t-p}] \\
&\hat{y}_{t|t-1} = \mathbb{E}\left[y_t | F_{t-1}\right] = f\left(Z_{t-1}, W\right) \\
&\text{then } \hat{Z}_{t|t-1} = \tilde{f}(Z_{t-1}, W) \\
&\hat{y}_{t+1|t-1} = \mathbb{E}\left[y_{t+1} | F_{t-1}\right] = \mathbb{E}\left[f\left(Z_t, W\right) | F_{t-1}\right] = \mathbb{E}\left[f\left(\tilde{f}(Z_{t-1}, W) + u_t, W\right) | F_{t-1}\right]
\end{aligned}
$$

Ideally one would need to estimate the last expectation term by numerical integration. However, moving beyond 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 as well as estimating this integral 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.[4] 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.

[4] 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 forecast is constructed as a simple average over the n previous periods (Stock & Watson (2007); or Aparicio & Bertolotto (2017)).

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

We tried n=3,6 or 12 and obtain very similar results in terms of model comparison.

2. (V)AR(p) model:

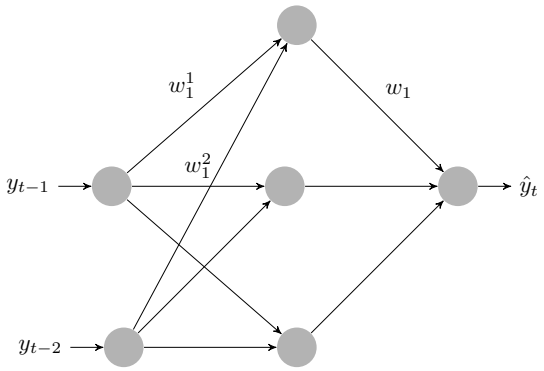
$$\hat{y}_{t|t-1} = A + BZ_{t-1}, \text{with } Z_{t-1} = [y_{t-1}, ...y_{t-p}]$$

3. NN (simple fully-connected Neural Network)

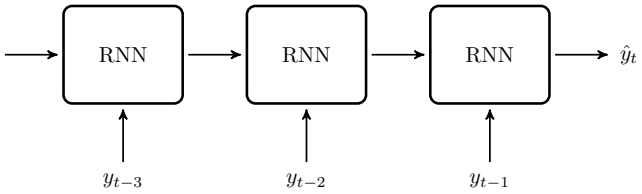
$$\hat{y}_{t|t-1} = b + \sum_{n=1}^{N} w_n \cdot \sigma\big(b^n + \sum_{\tau=1}^{\mathrm{P}} w_\tau^n y_{t-\tau}\big) \tag{1}$$

where

- $\hat{y}_t$: Prediction for time t
- $y_{t-\tau}$: Previous time steps
- w_n: Weight from hidden unit n to the output
- w_τ^n: Weight from lag τ to hidden unit n
- b, b^n: Bias of output and hidden unit n
- σ: Non-linear function (ReLU)



4. Long short-term memory (LSTM) recurrent neural network (RNN). The representation of the standard RNN is given on the following figure.



The structure of the LSTM network is similar to the RNN except it has additional "gates". These gates allow the network to decide on its own what part of the network state it wants to remember on the next iteration and what part it can forget (Hochreiter & Schmidhuber (1997)).

The networks are trained by adaptive stochastic gradient descent optimizer - Adam (Kingma & Ba (2014)). The difference of Adam compared to the standard stochastic gradient descent algorithm is that Adam updates each parameter separately and that it changes the speed of adjustment depending on the "momentum" m_t or approximated first-oder moment and the "friction" v_t or approximated second-order moment of the gradient.

$$
\begin{aligned}
m_t &= \beta_1 m_{t-1} + (1-\beta_1)\cdot g_t \\
v_t &= \beta_2 v_{t-1} + (1-\beta_2)\cdot g_t^2 \\
\hat{m}_t &= \frac{m_t}{1-{\beta_1}^t} \\
\hat{v}_t &= \frac{v_t}{1-{\beta_2}^t} \\
\theta_{t+1} &= \theta_t - \eta \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon}
\end{aligned}
$$

Typical values: $\beta_1 = 0.9, \quad \beta_2 = 0.999$

2.2 Data

We use monthly data on annual CPI inflation for the US from 1960-01-01 until 2018-07-01 (703 observations), NA, in levels, from FRED, Federal Reserve Bank of St. Louis.[5]

[5] Future experiments will include: monthly CPI inflation, multivariate forecasts, PCE inflation

3 Results

Table 1 presents the mean and standard deviations of RMSFE of all forecasting models relative to RW. Each model's hyper-parameters are selected by Monte-Carlo cross validations. One can see that the numbers for AR model are very closet to 1 especially beyond the 1-step-ahead forecast. In other words its performance is very close to RW. The errors of the NN and the LSTM are smaller than 1. At 1-step-ahead horizon, the LSTM error is only a halve of the RW, and the RMSFE of the simple NN is 40% of the random walk's error. As the number of forecast steps h increases, the errors of the NN model becomes closer in magnitude to the RW forecast errors. However, even at the 12-step-ahead the NN performs slightly better than AR and RW. The most striking result is that LSTM outperforms all other models at longer horizons. The RMSFE of the LSTM is on average just one half of the RW's.

Table 1: CPI Forecast RMSFE relative to Random Walk

	Test Error	h=2	h=3	h=6	h=12
(V)AR	0.83 (0.11)	0.96 (0.12)	1.06 (0.14)	0.95 (0.11)	0.97 (0.10)
LSTM	0.55 (0.06)	0.54 (0.04)	0.50 (0.09)	0.43 (0.12)	0.56 (0.18)
NN	0.40 (0.03)	0.53 (0.04)	0.65 (0.05)	0.71 (0.04)	0.91 (0.03)
RW	0.66	0.79	0.93	1.36	2.03

Standard errors are computed by 20 runs of Monte-Carlo cross-validation. The values for RW are absolute. h indicates the forecast horizon.

Since our networks are trained by a local optimizer multiple optima might be a concern. We follow the literature (for example Stock & Watson (1998)) and look at the distributions of the forecast errors after training the networks on the same train and test set but starting them at different initial conditions. Low multi-start variance (the second number in brackets in Table 2) indicate that the NN and the LSTM converge approximately to the same forecast (in terms of RMSFE) in each optimization chain.

Table 2: CPI Forecast RMSFE

Model	Test Error	h=2	h=3	h=6	h=12
(V)AR	0.54 (0.08)	0.76 (0.09)	0.99 (0.13)	1.30 (0.15)	1.96 (0.20)
LSTM	0.36 (0.04, 0.01)	0.43 (0.03, 0.00)	0.46 (0.08, 0.00)	0.58 (0.16, 0.01)	1.13 (0.37, 0.03)
NN	0.27 (0.02, 0.00)	0.42 (0.03, 0.00)	0.60 (0.04, 0.00)	0.97 (0.05, 0.01)	1.85 (0.05, 0.01)

Mean and standard deviation of various models after 20 runs of Monte-Carlo cross-validation and after 20 runs with randomized starting values.

4 Sensitivity Analysis

As we discuss in the introduction the performance of different NN models is crucially dependent on the selected hyper-parameters. It can also depend on the lag selection criterion and max number of lags that this criterion is allowed to choose from. To access the sensitivity of our results we train our forecasting model with all possible parameter combinations and rank them based on the 1-step-ahead test set error. Table 3 presents the share of the top 10 performers that is represented by the corresponding model. For example, at 1-step-ahead horizon all the best 10 models are NNs. This means that NNs are in general better for short term forecasting and that they are robust to different hyper-parameter selections. LSTM models, on the contrary are rather sensitive to their tuning parameters. Only a small portion of LSTMs enter the top list. Even though for h =6 to 12 the top LSTM model is better than the top NN (as can be seen in the previous Table) the second and third best LSTMs are closely followed by many NN models.

This can also be observed in Tables 5 and 6 which present the top 10 NNs and top 10 LSTM specifications respectively. While all NN models are very close in terms of their test errors, the performance of the LSTM is affected significantly by the selected hyper-parameters.

As can be seen in Table 5 and 6 the simple NN performs 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 lags. The results for the RNN are rather mixed with regard to the information criteria and the maximum number of lags. The best performing model is the one with a fixed and large number of

lags.

Figure 1 additionally plots the test errors of the NN (top row) and LSTM (bottom row) as a function of the number of hidden units, the initial leaning rate and the maximum number of epochs for which the network can be trained. 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 and for both models it is essential to allow for a large number of training epochs.

Table 3: Models Ranking: Fraction among the Top 10

	h=1	h=2	h=3	h=4	h=5	h=6	h=7	h=8	h=9	h=10	h=11	h=12
NN	10	9	9	9	9	7	7	7	7	6	6	4
(V)AR	0	0	0	0	0	0	0	0	0	0	0	1
LSTM	0	1	1	1	1	3	3	3	3	4	4	5

Table 4: Top Best Models of (V)AR with Parameters

	model	Test Error	h=2	h=3	h=6	h=12	infc	p	Lag	α
1	(V)AR	0.54	0.76	0.99	1.30	1.96	bic	2	12	0.9
2	(V)AR	0.55	0.74	0.98	1.32	1.85	bic	2	12	0.9
3	(V)AR	0.63	0.74	0.85	1.21	1.98	bic	14	20	0.9
4	(V)AR	0.72	0.87	1.04	1.41	2.15	None	20	20	0.9

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, α - share of the data used as a train set.

5 Layer-wise Relevance Propagation

After fitting NN and LSTM models one might be interested in interpreting what the networks learn from the data. In contrast to linear models it is not possible to directly interpret the weights of the NNs since the final prediction is a nonlinear function of the network parameters. Lapuschkin *et al.* (2016) suggest to access the importance of the model inputs by LRP. This procedure attaches a value to every neuron (including the

Table 5: Top Best Models of NN with Parameters

	model	Test Error	h=2	h=3	h=6	h=12	n	infc	p	Lag	LR	n epochs	α
1	NN	0.27	0.42	0.60	0.97	1.85	100	bic	14	20	0.001	1000	0.9
2	NN	0.27	0.44	0.62	1.00	1.86	20	bic	14	20	0.01	1000	0.9
3	NN	0.27	0.43	0.60	0.97	1.85	100	bic	14	20	0.001	500	0.9
4	NN	0.28	0.43	0.60	0.95	1.86	100	bic	14	20	0.01	500	0.9
5	NN	0.28	0.44	0.62	0.97	1.87	100	bic	14	20	0.01	1000	0.9
6	NN	0.30	0.49	0.64	0.99	2.28	10	bic	14	20	0.10	1000	0.9
7	NN	0.31	0.49	0.66	1.10	1.73	100	aic	19	20	0.01	1000	0.9
8	NN	0.31	0.54	0.69	1.11	3.29	100	bic	14	20	0.10	1000	0.9
9	NN	0.32	0.51	0.73	1.21	2.08	100	None	20	20	0.001	1000	0.9
10	NN	0.32	0.47	0.68	1.16	3.25	10	aic	19	20	0.10	1000	0.9

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, α - share of the data used as a train set.

Table 6: Top Best Models of LSTM with Parameters

	model	Test Error	h=2	h=3	h=6	h=12	n	infc	p	Lag	LR	n epochs	α
1	LSTM	0.36	0.43	0.46	0.58	1.13	100	None	20	20	0.10	1000	0.9
2	LSTM	0.37	0.59	0.75	1.01	1.54	20	aic	19	20	0.01	1000	0.9
3	LSTM	0.49	0.68	0.81	1.05	1.69	20	bic	14	20	0.10	1000	0.9
4	LSTM	0.51	0.68	0.83	1.25	1.99	20	None	12	12	0.10	1000	0.9
5	LSTM	0.68	0.80	0.95	1.37	2.12	100	None	12	12	0.01	100	0.9
6	LSTM	0.69	0.77	0.90	1.35	2.07	100	None	12	12	0.001	100	0.9
7	LSTM	0.71	0.81	0.94	1.08	1.75	100	hqic	16	20	0.10	100	0.9
8	LSTM	0.78	0.90	1.01	1.26	1.83	20	bic	14	20	0.001	500	0.9
9	LSTM	0.83	0.93	1.05	1.43	2.05	20	None	12	12	0.10	500	0.9
10	LSTM	0.83	0.94	1.26	1.72	2.53	100	bic	2	12	0.10	100	0.9

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, α - share of the data used as a train set.

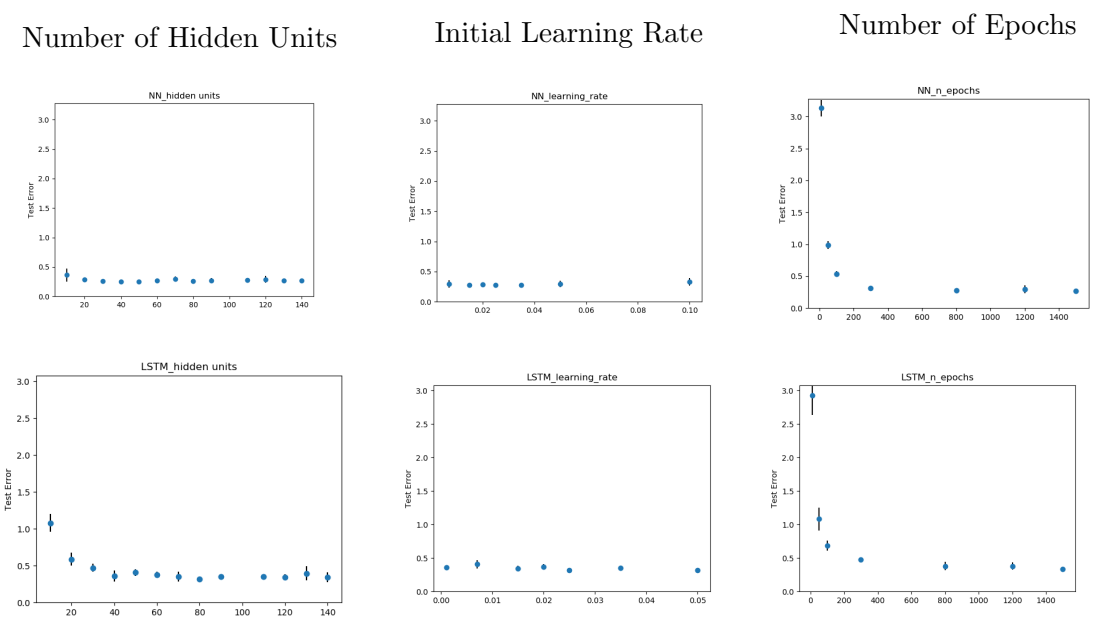


Figure 1: Test Error of different parameter choices for NN and LSTM

input neurons - lags) that quantifies its contribution to the network output. This value is called "relevance". It is computed layer-wise by smartly aggregating the signals that the neuron contributes to its successors.

Figure 2 depicts two examples the LRP measure for NNs (top row) and LSTM (bottom row). Blue slid line is the real data. Each bar represents the relevance of a particular lag which was fed into the model as an input (14 lags in total). The rightmost value of each plot is the 1-step-ahead prediction of the corresponding network. The dark red colors denote high relevance whereas white represents zero relevance.

As we can see most important for the NNs' and RNNs' predictions are the most recent lags (t=-1). Moreover the 12th and 13th lags contribute a lot to the final network outcome, especially for NNs. Additionally, the lags with the kins in the data are sometimes important.

6 Real Time Forecasting

Finally we conduct a counter-factual exercise to answer the question "What would have happened if forecasters were using NN and LSTM models instead of AR(p) to

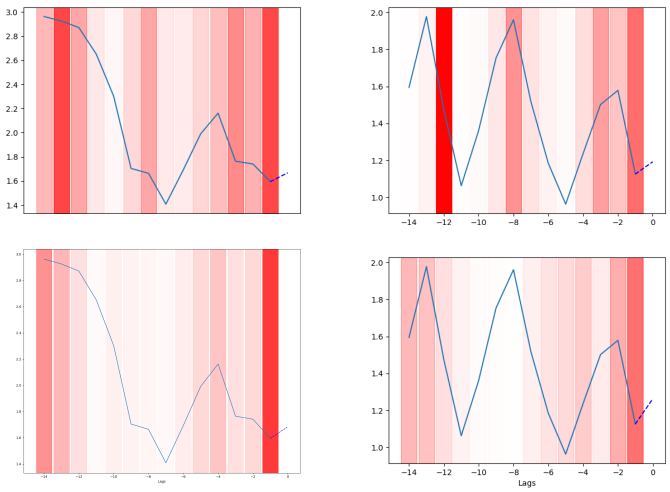


Figure 2: LRP plots for NN (top row) and LSTM (bottom row)

predict inflation since 2000?". We recursively compute inflation forecasts year by year. Estimation 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. Forecast for $h \geq 2$ are computed by iterating forward the 1-step-ahead forecast.

Figure 3 depicts several examples of the 12 month ahead predictions of the different models. Red solid lines depict the real data series and the blue dashed lines are the forecasts. From these pictures one can get some intuition why LSTM predictions for longer horizons are more accurate than the linear forecasts. The LSTM tends to represent a time series by a curve instead of a rising or falling line. This allows the LSTM to better fit unanticipated drops (as for example for the years 2009, 2010). It can also worsen the forecast if the true data was rather following a straight line (as for example in 2011). NNs produce nonlinear forecasts which are somehow closer to a straight line. It makes these forecast more accurate for shorter horizons. However, the forecast can become highly inaccurate at longer horizons.

We conduct the Diebold-Mariano Test (1995) with modification suggested by Harvey et. al (1997) to identify if the differences between three different forecasts are significant.[6]

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

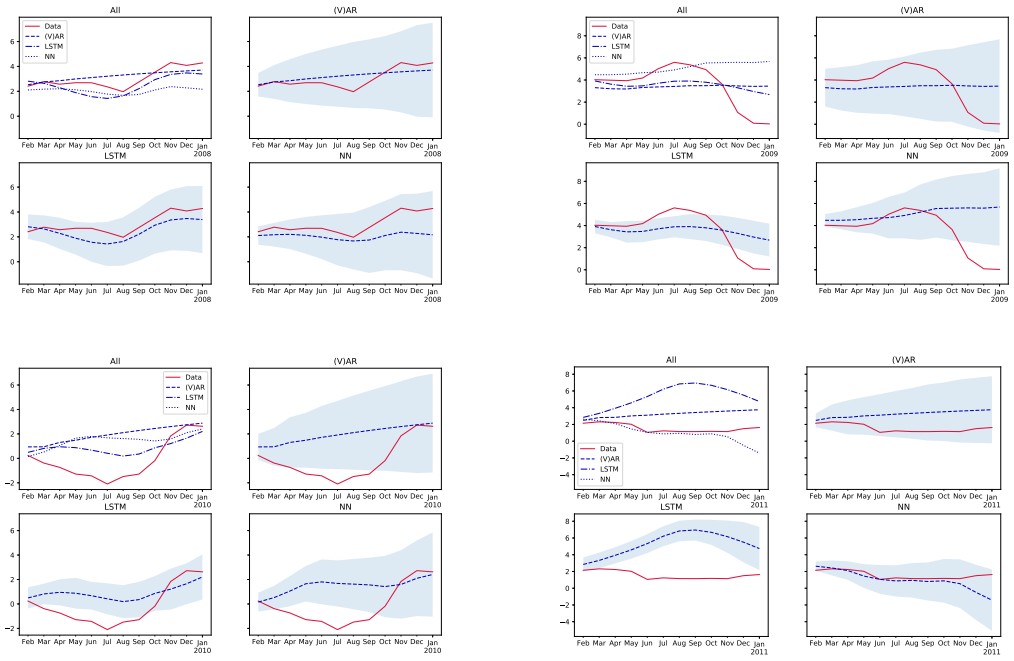


Figure 3: Real-time Forecast Examples

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.

Aparicio, Diego, & Bertolotto, Manuel. 2017. Forecasting inflation with online prices.

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.

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

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.

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.

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. 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.