Tölö, Eero

Working Paper

Predicting systemic financial crises with recurrent neural networks

Bank of Finland Research Discussion Papers, No. 14/2019

Provided in Cooperation with:
Bank of Finland, Helsinki

Suggested Citation: Tölö, Eero (2019) : Predicting systemic financial crises with recurrent neural networks, Bank of Finland Research Discussion Papers, No. 14/2019, ISBN 978-952-323-287-7, Bank of Finland, Helsinki, https://nbn-resolving.de/urn:nbn:fi:bof-201908271427

This Version is available at:
https://hdl.handle.net/10419/212448

Bank of Finland Research Discussion Papers
14 • 2019

Eero Tölö

Predicting systemic financial crises with recurrent neural networks

Bank of Finland
Research

Bank of Finland Research Discussion Papers
Editor-in-Chief Esa Jokivuolle

Bank of Finland Research Discussion Paper 14/2019
27 August 2019

Eero Tölö
Predicting systemic financial crises with recurrent neural networks

ISBN 978-952-323-287-7, online
ISSN 1456-6184, online

Bank of Finland
Research Unit

PO Box 160
FIN-00101 Helsinki

Phone: +358 9 1831

Email: research@bof.fi
Website: www.suomenpankki.fi/en/research/research-unit/

Predicting systemic financial crises with recurrent neural networks

Eero Tölö[*†]

Revised version

30 August 2019

Abstract:

We consider predicting systemic financial crises one to five years ahead using recurrent neural networks. The prediction performance is evaluated with the Jorda-Schularick-Taylor dataset, which includes the crisis dates and relevant macroeconomic series of 17 countries over the period 1870-2016. Previous literature has found simple neural network architectures to be useful in predicting systemic financial crises. We show that such predictions can be significantly improved by making use of recurrent neural network architectures, especially suited for dealing with time series input. The results remain robust after extensive sensitivity analysis.

JEL Classification: G21, C45, C52

Keywords: Early Warning System, Banking Crises, Neural Networks, Validation

[*] Department of Financial Stability, Bank of Finland.
Helsinki GSE, University of Helsinki, E-mail address: eero.tolo@helsinki.fi
Department of Economics, London School of Economics and Political Science.

[†] The work was conducted at the LSE during spring 2019. The author would like to thank Milan Vojnovic for useful comments and suggestions as well as anonymous comments received during the editorial process. The views expressed in the paper are solely those of the author and do not necessarily represent the views of the Eurosystem or the Bank of Finland.

1. Introduction

For obvious reasons, there is on-going interest to devise better models for forecasting macroeconomic turning-points such as onset of economic downturns, recessions, and systemic financial crises. This paper investigates the ability of different types of neural networks to predict systemic financial crises.

For purposes of this study, a systemic financial crisis can be thought merely as a recession amplified by a massive deleveraging of the financial sector (see Laeven and Valencia, 2012, Reinhart and Rogoff, 2009, and Jordà et al, 2015, for more precise definitions). Lists of crisis dates have been made widely available by previous research. Researchers have found several warning signs that predate such crisis events by a few years (see e.g. Alessi and Detken, 2011, Babecký et al, 2014, Barrell et al, 2011, Bordo and Meisner, 2012, Borio and Drehmann, 2009, Büyükkarabacak and Valev, 2010, Davis and Karim, 2008, Demirgüc-Kunt and Detragiache, 1998, 2000, Detken et al, 2014, Domaç and Martinez Peria, 2003, Drehmann and Juselius, 2014, von Hagen and Ho, 2007, Kaminsky and Reinhart, 1999, Kauko, 2014, Lo Duca and Peltonen, 2013 and Roy and Kemme, 2012, Tölö et al, 2018). These include a rapid increase in asset values, increased leverage of the private sector, and often current account deficit of a country that is borrowing from abroad to finance its excesses.

Earlier literature (see e.g. Alessi et al, 2014, Detken et al, 2014) has considered various crisis prediction methods ranging from variations of logistic models to decision trees and other machine learning methods. A few recent studies have also considered artificial neural networks (Fricke 2017, Holopainen and Sarlin, 2017, and Ristolainen, 2018) with mostly very promising results. The neural network approach may be justified on basis of nonlinear relationship between the indicators and the crisis event. For example, Kauko (2012) found that rapid credit growth predicted the deterioration of bank credit quality in the 2008 financial crisis only in countries that had a current account deficit.

So far in these studies, the neural network architectures have been limited to the multilayer perceptron with one hidden layer, which may not be optimal, given the time-series nature of the data. This study takes a step forward by considering how recent advances in the field of recurrent neural networks could benefit the predictions. Specifically, recurrent neural networks with parameter sharing enable the use of lagged values of predictors without introducing too many additional free parameters.

We consider three recurrent neural network architectures – a basic RNN, an RNN with long short-term memory (LSTM) cells, and an RNN with Gated Recurrent Units (GRUs), which we

benchmark against a multilayer perceptron model and the logistic regression. The data comes from the Jorda-Schularick-Taylor (JST) macrohistory database, which covers the financial crisis dates and relevant macroeconomics data for 17 advanced economies in the period 1870–2016.

We train the models to predict financial crises with a prediction horizon extending from 1 to 5 years. The models are subsequently tested out-of-sample either in a country-by-country cross-validation or a two-step sequential. We assess the out-of-sample prediction performance by the area under the ROC curve. We generally find that the recurrent neural network architectures outperform both the multilayer perceptron and the logistic regression benchmarks and exihibit robustness across different forecast horizons. The LSTM and GRU architectures have roughly equivalent performance. The results survive extensive sensitivity analysis.

Hence, our main contribution is to demonstrate the value of the recurrent neural networks for the task of systemic financial crisis prediction. Similar techniques could be useful in the related literature recession prediction, as in Qi (2001), and other types of crises, as in Fioramanti (2008), or more generally in predicting macroeconomic time series (see Smalter Hall and Cook, 2017).

The rest of the report is organized as follows. Section 2 offers a view on the scant literature of predicting systemic financial crises with neural networks. Section 3 reviews the neural network architectures considered in this study. Section 4 presents the data including the financial crisis dates. Section 5 describes the performance evaluation and validation frameworks. Section 6 presents the results from the performance evaluation and sensitivity analysis. How the results relate to earlier work is discussed in Section 7. Section 8 concludes.

2. Literature on predicting systemic financial crises with neural networks

Even though neural networks have been considered in various early warning systems, a thorough searched brought up only a handful of earlier studies that have considered neural networks in prediction of systemic financial crises. Fricke (2017) and Holopainen and Sarlin (2017) benchmark the crisis prediction performance among various machine learning methods including neural networks. Ristolainen (2018) focuses strictly on the neural network approach and benchmarks it against logistic regression. All four studies consider the most straightforward neural network architecture called one hidden layer perceptron. However, their results differ due to differences in the datasets and in the evaluation approach.

Fricke (2017) considers earlier version of the JST dataset that covers 14 developed countries over maximum of 137 years (1870–2008, whereas in the present study the dataset covers 17 developed economies over the period 1870-2016). He considers a prediction horizon of one year and a

minimal set of input features (five lags of credit growth) and finds that the logistic regression has most robust prediction performance overall. In his study, a classification forest and the one-hidden-layer perceptron (with 3 or 5 nodes in the hidden unit) outperform the logistic regression for some validation samples, while K-nearest-neighbors (KNN), quadratic discriminant analysis (QDA), support vector machines (SVM), are consistently worse than the logistic regression.

Holopainen and Sarlin (2017) consider a quarterly dataset of 15 European Union countries. Also unbalanced, their data covers 26 to 35 years for each country in the period 1976–2012. They consider a flexible prediction horizon of 5 to 12 quarters and include 14 input features at the time of prediction (no lagged variables). In their study, a perceptron neural network with eight hidden units and KNN are among the top performing methods and both outperform logistic model by a wide margin in all evaluations.

Ristolainen (2018) considers monthly dataset of 18 countries obtained from Kaminsky (2006). His data covers on average 23 years for each country and extends from 1970/80 to 2003. He considers prediction horizon of 24 months and includes 13 input features. In his study, perceptron neural networks also outperform logistic regression in all evaluations. He finds that grouping of similar countries also improves the predictions. The performance of the neural network improves rapidly when the number of units in the hidden layer are increased from 5 to 10 but does not change much for higher number of neurons.

The difference in the outcomes in the latter studies compared to Fricke (2017) appear to be borne mainly from the minimal number of features included in Fricke's study. Other factors, which would likely become significant once more features are added, are the more extended period, 137 years, and heterogeneity of the countries, and possibly the use of multiple lagged values of the features present in Fricke (2017).

3. Neural network architectures

A. Feedforward neural networks

A feedforward neural network is the earliest and most straightforward type of artificial neural network. In this network, the input data enters only once from the input nodes (see the bottom of Figure 1) and moves through the hidden nodes and to the output node(s).

A multilayer perceptron consists of three or more layers: an input layer, one or more hidden layers, and an output layer. The basic architectural question for the multilayer perceptron is the number and width of the hidden layer(s). According to universal approximation theorem for neural networks (Hornik et al. 1989), every continuous function on a bounded domain can be

approximated with a multilayer perceptron with just one hidden layer. The problem is that the required size for such a network can be impractically large making the network prone to overfitting. Empirical evidence generally suggests that depth can be beneficial. However, as our primary interest in this study lies in the recurrent neural networks, we only consider the single hidden layer perceptron used in the earlier related work by Fricke (2018), Holopainen and Sarlin (2017), and Ristolainen (2018).

In our basic setup the number of nodes in the hidden layer is 10 and the number of input features is 5 (or 25 when we include 5 lagged values of input features). The one hidden layer perceptron used in this study can be defined recursively as:

$$\boldsymbol{h}(\boldsymbol{X}) = a_{relu}(\boldsymbol{W}\boldsymbol{X} + \boldsymbol{b}), \tag{1}$$

$$o(\boldsymbol{h}) = a_{sigmoid}(\boldsymbol{V}\boldsymbol{h} + \boldsymbol{c}), \tag{2}$$

where a are activation functions (applied elementwise), $\boldsymbol{W}$ and $\boldsymbol{V}$ are weight matrices, $\boldsymbol{b}$ and $\boldsymbol{c}$ are bias vectors, $\boldsymbol{X}$ is the input, and o is the output. Following common practice, we apply rectified-linear (relu) activation function at the hidden nodes and sigmoid activation at the single output node.

To manage the overfitting, we include L2 regularization, which is a squared penalty term for the neural network weights. The regularization parameter is initially set to 0.1 following Ristolainen (2018), while other values are included in the sensitivity analysis.

The weights and biases are optimized by minimizing a cross-entropy loss function that compares the predicted output to the actual known state (crisis or not). We train the neural network by Adam, an adaptive variation of the gradient descent algorithm utilizing the backpropagation algorithm for fast computation of gradients. Various other training algorithms and hyperparameters are considered in the sensitivity analysis. The weights and biases ($\boldsymbol{W}$, $\boldsymbol{V}$, $\boldsymbol{b}$, $\boldsymbol{c}$) are initially set at random values.

B. Recurrent neural networks

Recurrent neural networks (RNNs) are a family of neural networks designed for sequential data such as language and time series. The RNN accepts input data sequentially, which allows RNNs to use their hidden states (akin to memory) dynamically to process a sequence of input data. A key idea is parameter sharing, which restricts the number of parameters in the model and helps avoid overfitting. For example, Smalter Hall and Cook (2017) use recurrent neural networks to predict unemployment.

In this study, we consider three different RNN architectures: basic RNN, RNN with Long-Short Term Memory (LSTM) cells, and RNN with GRU cells.

In the basic RNN presented in Figure 2, there is a hidden state $\boldsymbol{h}_t$ of dimensionality h, which evolves through a number of time steps T. The evolution of $\boldsymbol{h}_t$ depends on the previous hidden state $\boldsymbol{h}_{t-1}$ and the current input $\boldsymbol{X}_t$. At the final time step, the hidden state is mapped to output and the weights and biases of the network are again trained to minimize a cross-entropy loss function. In our main results, the dimensionality of the hidden state is 10, and the number of time steps is 5.

The basic RNN can be defined recursively as

$$\boldsymbol{h}_t(\boldsymbol{h}_{t-1}, \boldsymbol{X}_t) = a_{tanh}(\boldsymbol{W}\boldsymbol{X}_t + \boldsymbol{U}\boldsymbol{h}_{t-1} + \boldsymbol{b}), \qquad t = 1, 2, 3, \ldots, T\ , \tag{3}$$

$$o(\boldsymbol{h}_T) = a_{sigmoid}(\boldsymbol{V}\boldsymbol{h}_T + \boldsymbol{c})\ . \tag{4}$$

The network is trained similarly as the feedforward neural network: the output enters a cross-entropy loss function that is minimized with algorithm from the gradient descent family of methods. However, it should be noted that due to repeated multiplication of the hidden state by the same $\boldsymbol{U}$, the basic RNNs are susceptible to the problem of vanishing or exploding gradient. LSTM type RNNs were developed to deal with this problem.

The idea of an LSTM was proposed by Hochreiter and Schmidhuber (1997) and it has turned out to be quite popular. The idea is to make the recurrence going from $\boldsymbol{h}_t$to $\boldsymbol{h}_{t+1}$ more complicated such that the network can control what kind of information propagates onward from one time step to another. To visualize the LSTM network, think of each hidden node in Figure 2 being replaced by an LSTM cell depicted in Figure 3. The hidden state is now composed of two components $\boldsymbol{h}_t$ and $\boldsymbol{s}_t$, which both have dimensionality h. Gating units σ are elementwise sigmoid-functions that control the flow of information at points (x), x denoting Hadamard product ⊙ (elementwise multiplication).

We denote by $\boldsymbol{f}$, $\boldsymbol{i}$, and $\boldsymbol{o}$ the results from forget, input, and output gate (not to be confused with the output layer of the neural network) respectively. The LSTM cell operation can then be written as:

$$\boldsymbol{f}_t = \sigma(\boldsymbol{W}^f \boldsymbol{x}_t + \boldsymbol{U}^f \boldsymbol{h}_{t-1} + \boldsymbol{b}^f), \tag{5}$$

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}^i \boldsymbol{x}_t + \boldsymbol{U}^i \boldsymbol{h}_{t-1} + \boldsymbol{b}^i), \tag{6}$$

$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}^o \boldsymbol{x}_t + \boldsymbol{U}^o \boldsymbol{h}_{t-1} + \boldsymbol{b}^o), \tag{7}$$

$$\boldsymbol{s}_t = \boldsymbol{f}_t \odot \boldsymbol{s}_{t-1} + \boldsymbol{i}_t \odot a_{tanh}(\boldsymbol{W}\boldsymbol{x}_t + \boldsymbol{U}\boldsymbol{h}_{t-1} + \boldsymbol{b}), \quad (8)$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot a_{tanh}(\boldsymbol{s}_t). \quad (9)$$

In the main results, the dimensionality of the hidden state is 10, and the number of time steps is 5. Alternative hyperparameters are considered in the sensitivity analysis. Training the LSTM network is similar to training a feedforward neural network as described previously. To manage overfitting, we initially use the same L2 regularization weight 0.1 for the recurrent neural networks as for the feedforward neural networks.

GRU is a gating mechanism proposed by Cho et al. (2014) with a similar purpose as the LSTM. It has only two gates - a reset gate and an update gate - and a single vector presents the hidden state. Hence, it has somewhat fewer parameters than the LSTM, so it is computationally more efficient, but LSTM cells can do more complex tasks than GRU cells. However, GRU has been shown to exhibit better performance in some relatively small datasets.

Following update equations describe GRU cell:

$$\boldsymbol{u}_t = \sigma(\boldsymbol{W}^u\boldsymbol{x}_t + \boldsymbol{U}^u\boldsymbol{h}_{t-1} + \boldsymbol{b}^u), \quad (10)$$

$$\boldsymbol{r}_t = \sigma(\boldsymbol{W}^r\boldsymbol{x}_t + \boldsymbol{U}^r\boldsymbol{h}_{t-1} + \boldsymbol{b}^r), \quad (11)$$

$$\boldsymbol{h}_t = \boldsymbol{u}_t \odot \boldsymbol{h}_{t-1} + (1 - \boldsymbol{u}_t) \odot a_{tanh}(\boldsymbol{W}\boldsymbol{x}_t + \boldsymbol{U}(\boldsymbol{r}_t \odot \boldsymbol{h}_{t-1}) + \boldsymbol{b}). \quad (12)$$

In the main results, the dimensionality of the hidden state is 10, and the number of time steps is 5. Training the GRU follows the same steps as described previously.

4. Data

All input data for this study come from the Jorda-Schularick-Taylor macro history database (Jorda et al, 2017, and Knoll et al, 2019). The dataset includes 17 countries: Australia, Belgium, Canada, Switzerland, Germany, Denmark, Spain, Finland, France, UK, Italy, Japan, Netherlands, Norway, Portugal, Sweden, and the USA.

The target variable that we try to predict is the systemic financial crisis dummy variable, which takes value 1 for a year that marks the start of a systemic financial crisis in a given country. The dataset classifies financial crises as "events during which a country's banking sector experiences bank runs, sharp increases in default rates accompanied by large losses of capital that result in public intervention, bankruptcy, or forced merger of financial institutions" (Schularick and Taylor, 2012). Table 1 lists the crisis dates. We can infer from Table 1 that financial crises were

fairly common until the WW2. The WW2 was followed by a long financial calm until the crises started happening again starting from the 70s. This pattern discussed more extensively in Schularick and Taylor (2012) motives us to consider training for the full-sample, post-WW2 and the post-Bretton-Woods era (the 1970s onwards).

We consider following input variables that we call features according to the neural network nomenclature.

1. Loans to non-financial private sector divided by GDP, 1 year growth
2. Current account-to-GDP ratio, level
3. Real GDP, 1 year growth
4. Real house prices, 1 year growth
5. Real stock prices, 1 year growth

(6.) Consumer price index, 1 year growth

(7.) Public sector debt-to-GDP ratio, 1 year growth

(8.) Real loans to non-financial private sector, 1 year growth

(9.) Real loans to households, 1 year growth

(10.) Real loans to businesses, 1 year growth

(11.) Mortgage loans, 1 year growth

(12.) Short-term interest rate, level

(13.) Long-term interest rate, level

Our main result are calculated with neural networks that take features 1-5 as input. For models other than non-recurrent neural networks we additionally consider adding 5 lags of each feature as in Schularick and Taylor (2012) and in Fricke (2017). In the sensitivity analysis, we consider using all the 13 features, which corresponds to similar amount of features as in Ristolainen (2018) and Holopainen and Sarlin (2017).

5. Performance evaluation

Target variable

The target variable, the financial crisis dummy, is dealt with similarly as in Fricke (2017) apart from the fact that we additionally consider longer prediction horizons. For example, if the crisis happens in year 2007, then a one-year forecast based on 1 lagged value uses the indicator data

from 2006 only; a two-year forecast based on 1 lagged value uses the indicator data from 2005 only; and so on. Correspondingly, one-year forecast based on 5 lagged values uses the indicator data from the years 2002–2006; respective two-year forecast uses data from the year 2001–2005; and so forth.

Also note that the crises are considered as breaks in a series in the sense that in the above example, in the case of one-year forecast we do not attempt to predict a crisis in 2008 based on 2007 data if there was a crisis in 2007. Also, once a crisis has occurred, we do not try predict another until 5 years have passed. Similarly, in the two-year forecast, we do not attempt to predict a crisis "too late", e.g. in the case of two-year forecast of 2007 crisis, the prediction that would be based on base 2006 data is left outside the evaluation. This handling of the pre-crisis and post-crisis periods is largely similar to Sarlin and Holopainen (2017) and Ristolainen (2018).

However, compared to Sarlin and Holopainen (2017) and Ristolainen (2018), we treat forecast horizons somewhat differently as the RNN structure naturally facilitates the lagged values of features. For example, Ristolainen (2018) uses a continuous 24-month prediction horizon, and Sarlin and Holopainen (2017) consider 5–12 quarter horizon. In that case, the whole 24-month or the 5–12 quarter horizon is set as a pre-crisis target variable (=1). As Ristolainen (2018) points out, the pre-crisis dummy approach is widely applied with quarterly and monthly data to avoid considering large number of lagged predictors.

Cross-validation and sequential-validation

We consider two alternative out-of-sample performance evaluation frameworks, cross-validation and sequential-validation.

In the cross-validation, we exclude each country in turn, train the network, and then perform the out-of-sample prediction for each year for the country that was excluded. Then we pool all the predictions together and evaluate the AUC statistics. The cross-validation does not fully preserve temporal ordering in the sense that full information about the other countries are used to do prediction for the excluded country. However, because temporal ordering for the predicted country is preserved (as opposed to randomized cross-validation) and because all the input variables are transformed stationary, the validation can be considered quite robust.

```
Cross-validation algorithm pseudocode:
[1]: Loop C over countries:
[2]:        Train model excluding the data of country C
[3]:        Test model using data of country C only
[4]:        Store  prediction probabilities for country C.
[5]: end loop
[6]: Calculate the out-of-sample AUC by pooling the prediction probabilities.
```

In the sequential-validation, we split the sample into two parts. The earlier part is used for training and the latter part is used for testing. In this case, the temporal structure is fully preserved as we do not use any future information for prediction. The downside is that for practical sample splits, we are mainly predicting the 2007-2008 financial crisis.

Comparing the two validation methods, cross-validation is computationally more intensive as we have to train the neural network separately when each country is excluded. The benefit is that we get lower variance in the result, both due to being forced to train several times but also due to being able to use more data in the validation phase.

Performance measure

We evaluate the prediction performance based on the area under the ROC curve (AUC). Each neural network outputs a number that can be interpreted as a probabilistic forecast for the 0-1 crisis event.

If this probability is larger than some threshold h, then we say that the neural network predicts a crisis. Otherwise, the prediction is that there is no crisis. Correctly predicted crisis is labeled a true positive (TP). Correctly predicted tranquil state is labeled a true negative (TN). A false alarm is labeled a false positive (FP) and a missed pre-crisis state is labeled false negative (FN).

Sensitivity is defined as TP/(TP+FN) and specificity is defined as TN/(TN+FP). If we plot sensitivity vs. 1-specificity for all possible threshold values h we get the receiver operating characteristic (ROC) curve. The area under the ROC curve is an approximately proper scoring rule for the classification task. The best value of the AUC, one, is achieved for a perfect model that is able to distinguish the two states for some threshold h perfectly. A random guess obtains AUC = 0.5. For the AUC, a higher value is better.

6. Results

The presentation of results is structured in two sections. Section A presents the performance results first for the cross-validation and then for the sequential-validation. Section B presents sensitivity analysis, which, for brevity, focuses on the LSTM neural network. Similar sensitivity analysis for the other neural network architectures is available in the annex at the end of the report. A reader who wishes to quickly get the overall picture may now jump directly to the validation summary and Table 5.

A. Prediction performance

Cross-validation

Recall that in the cross-validation, each country, in turn, is used as the test sample while the other countries form the training sample. The less interdependent the time series of different countries, the more robust the cross-validation becomes.

Start by considering forecasting a financial crisis one year ahead. We consider three alternative time periods for the cross-validation: the full-sample from 1870 to 2016, the post-world-war II period from 1946 to 2016, and the post-Bretton-Woods period ranging from 1970 to 2016. The AUC evaluated for each prediction model for each period is shown in Table 2. In this test RNN-GRU performs best followed closely by the RNN-LSTM. Also, the logit model with 5 lags, the one-hidden layer perceptron with 5 lags, and the basic RNN outperform the corresponding single lag models. For example, for the post-Bretton-Woods period, the RNN-GRU achieves AUC = 0.801 while the logit model with 5 lags only attains AUC = 0.649. The perceptron model with five lags also consistently outperforms the logistic benchmark.

Let us now consider alternative forecast horizons up to five years. For brevity, we only consider the post-Bretton-Woods period but expect the results to be similar also for the other periods. Table 3 presents the AUC evaluated for each model and forecasting horizon. Again, RNN-LSTM and the RNN-GRU emerge as the most reliable prediction models and consistently outperform the logistic benchmarks as well as the one-hidden-layer perceptron models. For the prediction horizons from 2 to 4 years, the RNN-LSTM has slightly better score than the RNN-GRU.

Sequential validation

Next, we turn to the sequential evaluation. We train the model using data from some initial year, T0, until 2002 and then assess the prediction performance over the remaining period 2003-2016. We consider three alternative training periods T0 = 1870, 1946, and 1970, corresponding to the largest available sample, post-WW2 period and post-Bretton-Woods period, respectively.

Table 4 presents the prediction performance in the sequential evaluation for different prediction horizons. The results for each training sample are shown in panels a through b. In each table, we see again that RNN-LSTM and RNN-GRU tend to outperform the other methods by a significant margin. Moreover, the results tend to improve progressively when the training data is more recent, in other words, when we move from panel a to panel b and from panel b to panel c.

Compared to the cross-validation presented in previous tables, there is somewhat more variability in the results. Besides the strict temporal ordering imposed by the sequential-validation, the

increased variability is partly a result of running the neural network optimization only once here vs. 17 times in the cross-validation.

Validation summary

Overall, we have seen that the recurrent neural network architectures (RNN, LSTM, GRU) performed better than the one-hidden layer perceptron in both cross-validation and sequential validation. To summarize the results, following Ristolainen (2018) we produce Table 5 that shows the in-sample performance, the (cross-)validation performance[‡], and the (sequential) out-of-sample performance as measured by the AUC. The reported numbers are based on the post-Bretton Woods period data.

According to our understanding, the hidden-layer perceptron with multiple (5) lags falls behind primarily, because despite the L2 regularization weight, the model still becomes somewhat too overfitted during the training. The recurrent architectures avoid this overfitting by reliance on parameter sharing. Another important observation is the stronger performance of RNN-LSTM and RNN-GRU as compared to the basic RNN. We attribute this *both* to the fact that the LSTM/GRU avoid the exploding gradient problem that can sometimes hamper the performance of basic RNN, *and* to the enhanced ability (due to the memory state) of the LSTM/GRU networks to keep track of intertemporal dependencies. These and other issues are analyzed further in the following sensitivity analysis.

B. Sensitivity analysis

In this section, we do a sensitivity analysis based on neural network training and neural network model complexity. For brevity, the discussion here mainly focuses on the RNN-LSTM, which performed best in off-the-shelf evaluations presented in the previous section. Further sensitivity analysis for the other neural architectures can be found in the supplementary material. We first consider sensitivity to training and then sensitivity to complexity (=size) of the neural network. In the following analysis, we concentrate primarily on the sequential out-of-sample 3-year ahead prediction but we expect that the sensitivity results would be similar had we used some other forecast horizon or cross-validation.

Sensitivity to training

[‡] The cross-validation classification statistics are typically lower when using country-by-country cross-validation than when using a randomized cross-validation as in Holopainen and Sarlin (2017) and Ristolainen (2018).

As a non-convex optimization problem, training a neural network with a large number of parameters is highly non-trivial. Two optimization runs with different algorithms or different initial values hardly ever converge to the same result. Hence, the prediction performance tends to depend on how we train the network. The dependence on the initial values is easy to handle as we can train an ensemble of neural networks and take an average prediction. For the training process, besides the algorithm, considerations include different regularization techniques such L2 regularization and dropout, which are used to reduce overfitting. L2 regularization adds a penalty term proportional to the sum of the squared weights. Dropout randomly excludes neurons from the network in order to avoid excessive co-adaptation.

Figure 5 shows the sequential out-of-sample 3-year ahead prediction performance measured by AUC (using the 1970-2002 period for training) as a function of performed training epochs for different optimization algorithms using regularization weight 0.1 (Adam corresponds to what we did in the previous section). Panel a shows the result for a random seed and panel b shows the result for another random seed. Two important observations can be drawn. Despite some persistent performance differences, none of the algorithms strictly dominates the other algorithms. The performance typically reaches a maximum at a finite number of epochs and deteriorates after that. This deterioration seems to be a result of the network adapting too well to the training data, and it can be counteracted by increasing the regularization weights or introducing dropout.

Figure 6 panel a shows the effect of adding the dropout feature to the neural network, and panel b shows the effect of increasing the value of regularization weight to 0.3. Both adjustments seem to independently counteract the co-adaptation. After 500 epochs the average AUC among the algorithms is 0.797 with the dropout in panel a and 0.779 with the increased regularization weight. Even if single run for the algorithms is not statistically significant evidence, the results suggest that, in the case of LSTM, adding the dropout feature could be helpful and better option than increasing the regularization weight further.

As noted before, in the sequential evaluation the multilayer perceptron (with 5 lags) suffered from overfitting when regularization weight was set to 0.1. Increasing the regularization weight should alleviate this problem and possibly improve the prediction performance. Figure 7 illustrates the sensitivity of the neural network predictions to the strength of the L2 regularization weight parameter. Increasing the regularization weight significantly improves the performance of the 5-lag one-hidden layer perceptron. The 1-lag one-hidden layer perceptron improves moderately, while the RNNs perform best with little to no regularization. To check whether this affects our main conclusions, we calculate the sequential prediction performance for the hidden-layer perceptron as in Table 4a–c when we use L2 regularization weight 0.5 or 1.0. The results (available in supplementary material, Table S1, at the end of the document) show that while the

performance improves for the particular forecast horizon and training sample that was used to choose the value of the parameter, the improvement is less pronounced for other forecast horizons and training samples. Hence, the conclusions about the advantage of the RNN architecture remain valid.

Sensitivity to model complexity

As a final topic, we consider the sensitivity of the prediction results to the number of input features and the size of the neural network. Again the discussion focuses on the RNN-LSTM and the case of 3-year prediction based on the post-Bretton-Woods era sample.

The size of the RNN-LSTM neural network can be expanded by increasing the number of units in the hidden state. This leads to an increase in the number of weight parameters to be optimized. If we denote the number of output nodes in the LSTM cell by h and the number of input features by d, the number of parameters in the LSTM layer can be calculated as $4h^2+4(d+2)$. Also, there are h+1 parameters related to the single node output layer.

Figure 8 shows both the cross-validated and sequentially evaluated out-of-sample prediction performance for the RNN-LSTM neural network as a function of the number of nodes in the hidden state. Panel a presents the results with 5 input features and panel b presents the results with 13 input features. We see that in both panels a and b, increase in the dimension of the hidden state improves the prediction performance. Moreover, in cross-validation, the network using 13 input features tends to outperform the network using only 5 input features. However, in the sequential validation smaller number of input features is preferred.

Alternatively, we can expand the network by increasing the amount of input time steps, which corresponds to the number of times the recurrent neural network repeats the recurrent operation until the hidden state reaches the final output layer. Because of parameter sharing, this does not change the number of weight parameters to be optimized. Figure 9 shows the cross-validated prediction performance as a function of time steps. Both the sequential validation and cross-validation have the highest performance when the number of time steps is about 5.

7. Discussion

Our results relate to the earlier work that aims to predict systemic financial crises with neural networks reviewed in Section 2. Similar to Fricke (2018) and in contrast to Ristolainen (2018) and Holopainen and Sarlin (2017), we don't find very large performance difference between the logistic regression model and the multilayer perceptron. Three potential reasons are, first, the datasets differ, and second, Ristolainen (2018) and Holopainen and Sarlin (2017) use higher

frequency data and, lastly, they do not include lagged values of predictors. Additionally, Ristolainen (2018) mainly reports AUC for individual crises.

The AUC can't be compared directly to those in Ristolainen (2018) and Holopainen and Sarlin (2017) because of the above mentioned differences. However, compared Fricke (2018) whose dataset and methodology are similar to ours, our highest attained values of AUC are considerably higher.

In the cross-validation, we find the best forecast accuracy at the 3-year prediction horizon and using five time steps in the recurrent neural network. These numbers are consistent with the relatively long length of the financial cycle reported in the literature (8 to 20 years, see e.g. Borio 2014 or Filardo et al. 2018).

In future work, a straightforward extension would be to try datasets with quarterly or monthly frequency. Regarding the architecture, one could take the analysis a step further by considering slightly deeper network structures that allow for a higher level of abstraction. Additionally, it would make sense to try to pinpoint where the benefits of the neural network prediction come from, e.g. what are the non-linearities underlying the good out-of-sample performance.

8. Conclusions

Earlier literature has found artificial neural network with single hidden-layer perceptron to be excellent models for financial crisis forecasting. We have found that the prediction performance can be improved further with recurrent neural networks (RNNs) based on Long-Short Term Memory (LSTM) and Gated Recurrent Unit (GRU) cells. The capacity to handle time series data seems to be particularly important for the stable performance across different prediction horizons as compared to single layer perceptron architecture. These advantages derive from the parameter sharing inherent for the RNN structure, which efficiently counteracts overfitting. Overall, the results support the notion that recurrent neural networks could be a useful tool for time series applications in finance and empirical macroeconomics.

The recurrent neural network architectures are relatively straightforward to implement in practice using, for example, Python/Keras. The downside is that even if we were able to outperform the logistic regression practically without fine-tuning parameters, in general choosing the hyperparameters and training the model requires expertize and possibly considerable effort.

References

Alessi, L., Antunes, A., Babecky, J., Baltussen, S., Behn, M., Bonfim, D., Bush, O., Detken, C., Frost, J., Guimaraes, R., Havranek, T., Joy, M., Kauko, K., Mateju, J., Monteiro, N., Neudorfer, B., Peltonen, T. A., and Rusnak, M., Marques R., Paulo Manuel, Schudel, W., Sigmund, M., Stremmel, H., Smidkova, K., van Tilburg, R., Vasicek, B., Zigraiova, D., 2015, Comparing Different Early Warning Systems: Results from a Horse Race Competition Among Members of the Macro-Prudential Research Network. Available at SSRN: https://ssrn.com/abstract=2566165 or http://dx.doi.org/10.2139/ssrn.2566165

Alessi, L., Detken, K., 2011, Quasi real time early warning indicators for costly asset price boom/bust cycles: A role for global liquidity, European Journal of Political Economy, 27(3), pp. 520-533.

Babecký, J., Havránek, T., Matěju, J., Rusnák, M., Šmídková, K., Vašíček, B., 2014, Banking, debt and currency crises in developed countries: Stylized facts and early warning indicators, Journal of Financial Stability, 15, 1-17.

Barrell, R, Davis, E.P., Karim, D., Liadze, L., 2011, How idiosyncratic are banking crises in OECD countries? National Institute Economic Review, 216, R53-R58.

Bordo, M.D., Meissner, C.M., 2012, Does inequality lead to a financial crisis? Journal of International Money and Finance, 31, pp. 2147-2161.

Borio, C., 2014, The financial cycle and macroeconomics: What have we learnt?, Journal of Banking and Finance, Vol. 45, pp. 182–198.

Borio, C., Drehmann, M., 2009, Assessing the risk of banking crises – revisited. BIS Quarterly Review, March, pp. 29-46.

Büyükkarabacak, B., Valev, N.T., 2010, The role of household and business credit in banking crises, Journal of Banking and Finance, 34, pp. 1247-1256.

Cho, K.; van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y., 2014, Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1724–1734.

Davis, E.P., Karim, D., 2008, Comparing Early Warning Systems for Banking Crises, Journal of Financial Stability, 4(2), pp. 89-120.

Demirgüc-Kunt, A., Detragiache, E., 1998, The Determinants of Banking Crises in Developed Countries, IMF Staff Papers, 45(1), 81–109.

Demirgüc-Kunt, A., Detragiache, E., 2000, Monitoring Banking Sector Fragility: A Multivariate Logit Approach, World Bank Economic Review, 14, pp. 287-307.

Detken. K, Weeken, O., Alessi, L., Bonfim, D., Boucinha, M., Castro, C., Frontczak, S., Giordana, G., Giese, J., Jahn, N., Kakes, J., Klaus, B., Lang, J., Puzanova, N., Welz, P., 2014, Operationalising the countercyclical capital buffer: indicator selection, threshold identification and calibration options, ERSB Occasional Paper Series No. 5 / June 2014.

Domaç, I., Martinez Peria, M.S., 2003, Banking crises and exchange rate regimes: is there a link? Journal of International Economics 61.

Drehmann, M., Juselius, M., 2014, Evaluating early warning indicators of banking crises: Satisfying policy requirements, International Journal of Forecasting 30, pp. 759-780.

Filardo, A., Lombardi, M., Raczko, M., 2018, Measuring financial cycle time, BIS Working Papers 755.

Fioramanti, M., 2008, Predicting Sovereign Debt Crises Using Artificial Neural Networks: A Comparative Approach, Journal of Financial Stability, Vol. 4, No. 2.

Fricke, D., Financial Crisis Prediction: A Model Comparison (November 29, 2017). Available at SSRN: https://ssrn.com/abstract=3059052 or http://dx.doi.org/10.2139/ssrn.3059052.

von Hagen, J., Ho, T-K., 2007, Money market pressure and the determinants of banking crises. Journal of Money, Credit and Banking, 39, pp. 1037-1066.

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

Hornik, K., 1991, Approximation Capabilities of Multilayer Feedforward Networks, Neural Networks, 4(2), 251–257.

Holopainen, M., Sarlin, P. 2017, Toward robust early-warning models: a horse race, ensembles and model uncertainty, Quantitative Finance, Vol. 17, pp. 1933–1963.

Jordà, Ò., Schularick, M., and Taylor, A. M., 2015, Leveraged Bubbles, Journal of Monetary Economics 76, S1-S20.

Kaminsky, G., Reinhart, C., 1999, The twin crises: the causes of banking and balance-of-payments problems, IMF Staff Papers, 45.

Kauko, K., 2012, External deficits and non-performing loans in the recent financial crisis, Economics Letters, 2012, vol. 115, issue 2, 196-199.

Kauko, K., 2014, How to foresee banking crises? A survey of the empirical literature, Economic Systems, 38, pp. 289-308.

Knoll, K., Schularick, M., and Steger, T., 2016, No Price Like Home: Global House Prices 1870-2012, American Economic Review 107(2): 331-353.

Laeven, L., Valencia, F., 2012, Systemic Banking Crises Database: An Update, Working Paper No. 12/163.

Lo Duca, M., Peltonen, T., 2013, Assessing systemic risks and predicting systemic events, Journal of Banking and Finance, 37, pp. 2183-2195.

Olmedo, E., 2014, Forecasting Spanish Unemployment Using Near Neighbour and Neural Net Techniques, Computational Economics, vol. 43, no. 2, pp. 183–197.

Qi, M., 2001, Predicting US recessions with leading indicators via neural network models, International Journal of Forecasting, vol. 17, issue 3, 383-401.

Reinhart, C. M., Rogoff, K. S., 2009, This Time Is Different: Eight Centuries of Financial Folly, Princeton University Press.

Ristolainen, K., 2018, Predicting Banking Crises with Artificial Neural Networks: The Role of Nonlinearity and Heterogeneity, Scandinavian Journal of Economics, 120(1), 31–62.

Roy, S., Kemme, D.M., 2012, Causes of banking crises: Deregulation, credit booms and asset bubbles, then and now, International Review of Economics and Finance, 24, pp. 270-294.

Schularick, M., Taylor, A. M., 2013, Credit Booms Gone Bust: Monetary Policy, Leverage Cycles and Financial Crises, 1870-2008, American Economic Review, 102(2): 1029-1061.

Smalter Hall, A., Cook, T. R., 2017, Macroeconomic Indicator Forecasting with Deep Neural Networks, Federal Reserve Bank of Kansas City, Research Working Paper 17-11, September. Available at https://doi.org/10.18651/RWP2017-11

Tölö, E., Laakkonen, H., Kalatie, S., 2018, Evaluating Indicator for Use in Setting the Countercyclical Capital Buffer, International Journal of Central Banking, Vol. 14, 2, pp. 51–1.

Figure 1. Perceptron with one-hidden layer

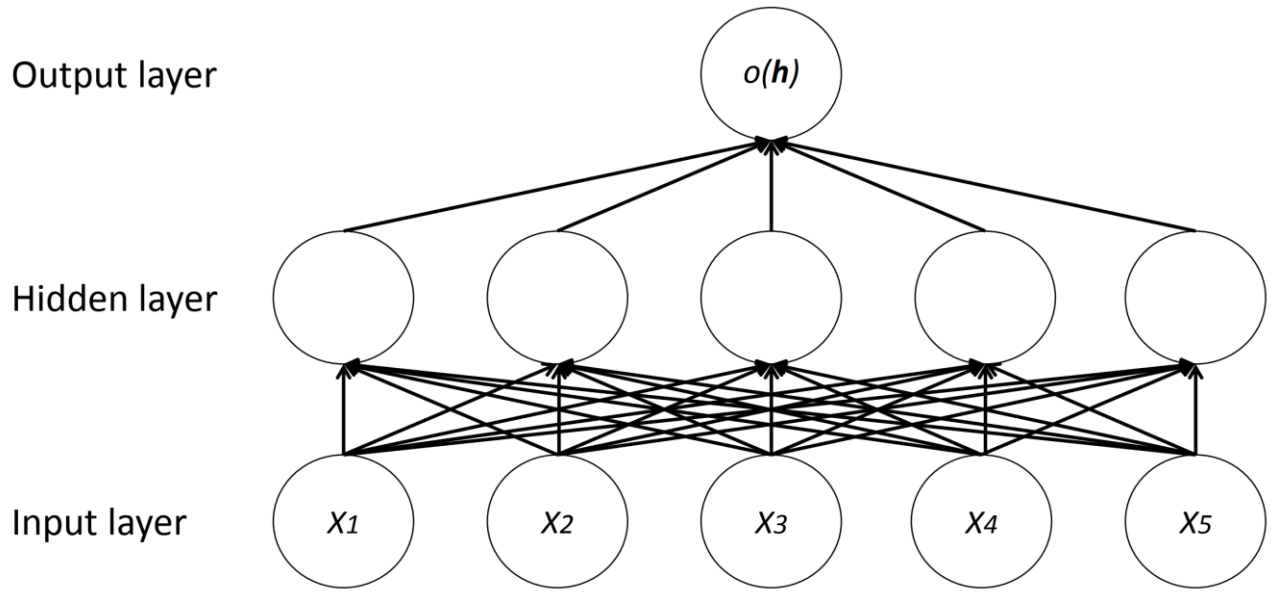


Figure 2. Basic recurrent neural network

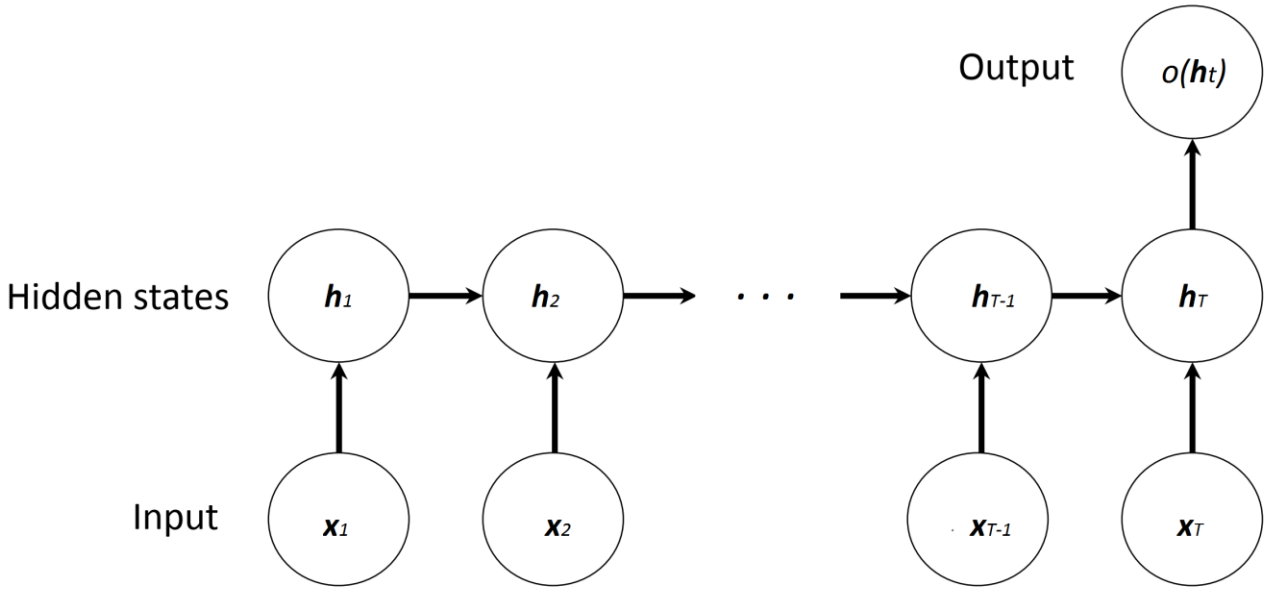


Figure 3. Long-Short Term Memory cell

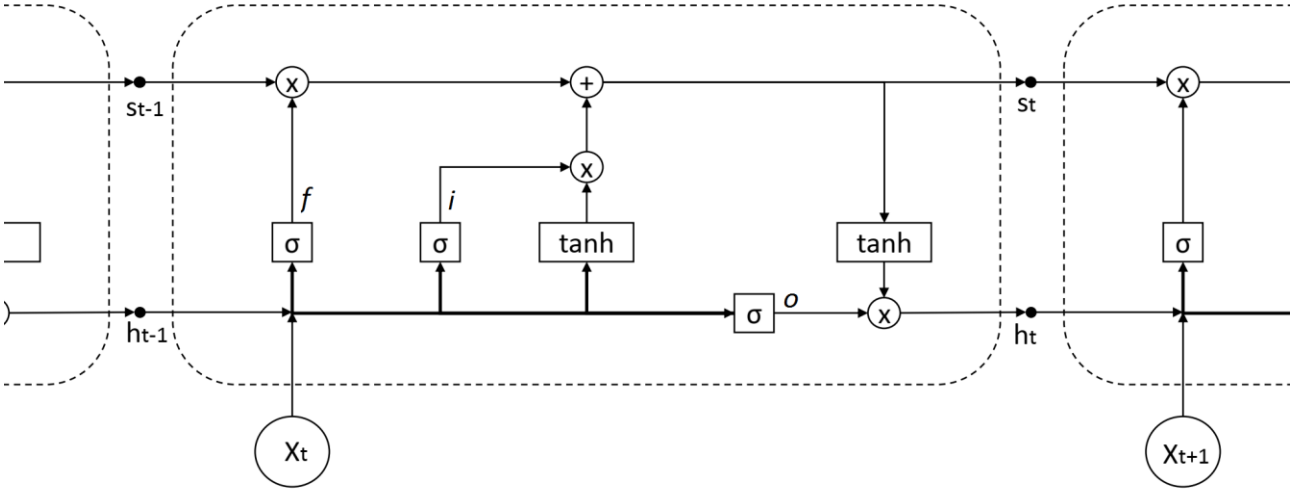


Figure 4. Gated Recurrent Unit

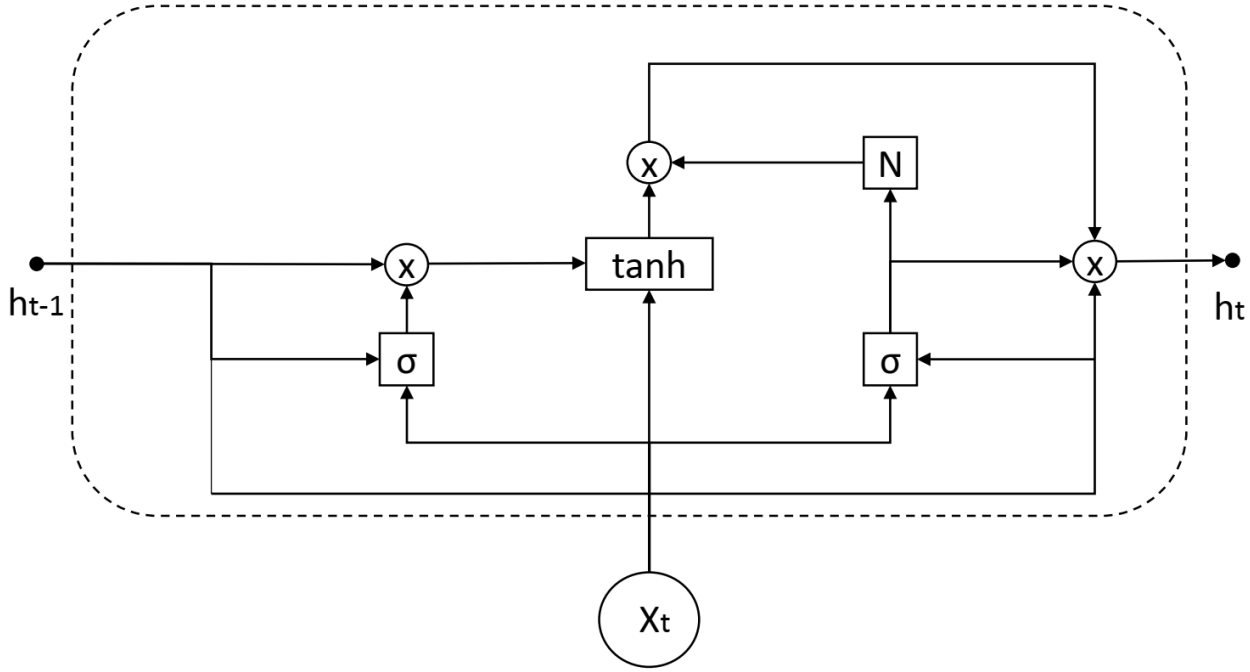


Table 1. Crisis dates in the Jorda-Schularick-Taylor database.

1870:	Belgium, Switzerland
1871:	Japan
1873:	Germany, Italy, USA
1877:	Denmark, Finland
1878:	Sweden
1882:	France
1883:	Spain
1885:	Belgium, Denmark
1887:	Italy
1889:	France
1890:	Japan, Portugal, Spain
1890:	UK
1891:	Germany
1893:	Australia, Italy, Netherlands, USA
1899:	Norway
1900:	Finland
1901:	Germany
1907:	Canada, Germany, Italy, Japan, Netherlands, Sweden, USA
1908:	Denmark
1910:	Switzerland
1913:	Spain
1920:	Japan, Portugal, Spain
1921:	Denmark, Finland, Italy, Netherlands
1922:	Norway, Sweden
1923:	Portugal
1924:	Spain
1925:	Belgium
1927:	Japan
1929:	USA
1930:	France, Italy
1931:	Belgium, Denmark, Finland, Germany, Norway, Portugal, Spain, Sweden, Switzerland
1934:	Belgium
1935:	Italy
1939:	Belgium, Netherlands
1974:	UK
1977:	Spain
1984:	USA
1987:	Denmark
1988:	Norway
1989:	Australia
1990:	Italy
1991:	Finland, Sweden, Switzerland, UK
1997:	Japan
2007:	UK, USA
2008:	Belgium, Denmark, France, Germany, Italy, Netherlands, Portugal, Spain, Sweden, Switzerland

Table 2. Cross-validation results with 1-year forecast horizon.

		Time period		
Model	Lags	Full sample	Post WW2	1970-2016
Logit	1	0.610	0.623	0.618
Logit	5	0.637	0.674	0.649
NN	1	0.607	0.607	0.608
NN	5	0.690	0.700	0.687
RNN	5	0.654	0.692	0.619
RNN-LSTM	5	0.714	0.774	0.761
RNN-GRU	5	**0.762**	**0.798**	**0.801**

Table notes. The numbers in the table are AUC. Number of features is 5, number of neural units in the hidden layer or RNN hidden state is 10, L2 weight is 0.1, training algorithm is Adam, training time is 100 epochs, minibatch size is 16. All the models use up to five lags of the same variables: real annual house price growth, real annual stock index growth, annual growth in credit-to-GDP ratio, current account-to-GDP ratio, and annual growth in real GDP.

Table 3. Cross-validation results for different forecast horizons.
Time period is 1970-2016.

		Forecast horizon				
Model	Lags	1	2	3	4	5
Logit	1	0.618	0.655	0.671	0.692	0.420
Logit	5	0.649	0.693	0.735	0.755	0.721
NN	1	0.608	0.729	0.700	0.687	0.552
NN	5	0.687	0.696	0.756	0.682	0.691
RNN	5	0.619	0.695	0.726	0.707	0.728
RNN-LSTM	5	0.761	**0.762**	**0.835**	**0.799**	**0.759**
RNN-GRU	5	**0.801**	0.750	0.783	0.795	0.750

Table notes. The numbers in the table are AUC. Number of features is 5, number of neural units in the hidden layer or RNN hidden state is 10, L2 weight is 0.1, training algorithm is Adam, training time is 100 epochs, minibatch size is 16. All the models use up to five lags of the same variables: real annual house price growth, real annual stock index growth, annual growth in credit-to-GDP ratio, current account-to-GDP ratio, and annual growth in real GDP.

Table 4. Sequential-validation results for different forecast horizons.
The training period is indicated in each panel and the test period is 2003-2016.

a) Training period 1870-2002

		Forecast horizon				
Model	Lags	1	2	3	4	5
Logit	1	0.535	0.616	0.744	0.757	0.417
Logit	5	0.505	0.521	0.671	0.671	0.633
NN	1	0.553	0.614	0.766	**0.760**	0.622
NN	5	0.363	0.431	0.542	0.408	0.632
RNN	5	0.557	0.616	**0.781**	0.701	0.656
RNN-LSTM	5	**0.584**	**0.742**	0.643	0.731	**0.850**
RNN-GRU	5	0.531	0.721	0.728	0.745	0.820

Table notes. The numbers in the table are AUC. Number of features is 5, number of neural units in the hidden layer or RNN hidden state is 10, L2 weight is 0.1, training algorithm is Adam, training time is 100 epochs, minibatch size is 16. All the models use up to five lags of the same variables: real annual house price growth, real annual stock index growth, annual growth in credit-to-GDP ratio, current account-to-GDP ratio, and annual growth in real GDP.

b) Training period 1946-2002

		Forecast horizon				
Model	Lags	1	2	3	4	5
Logit	1	0.558	0.602	0.748	0.672	0.458
Logit	5	0.426	0.41	0.513	0.575	0.622
NN	1	0.485	0.612	0.783	0.657	0.644
NN	5	0.415	0.448	0.238	0.357	0.751
RNN	5	0.551	0.641	**0.799**	0.685	0.694
RNN-LSTM	5	0.595	0.717	0.735	**0.786**	**0.873**
RNN-GRU	5	**0.609**	**0.744**	0.727	0.678	0.838

Table notes. The numbers in the table are AUC. Number of features is 5, number of neural units in the hidden layer or RNN hidden state is 10, L2 weight is 0.1, training algorithm is Adam, training time is 100 epochs, minibatch size is 16. All the models use up to five lags of the same variables: real annual house price growth, real annual stock index growth, annual growth in credit-to-GDP ratio, current account-to-GDP ratio, and annual growth in real GDP.

c) Training period 1970-2002

		Forecast horizon				
Model	Lags	1	2	3	4	5
Logit	1	0.574	0.608	0.713	0.685	0.338
Logit	5	0.398	0.366	0.566	0.698	0.736
NN	1	0.502	0.551	0.732	0.674	0.479
NN	5	0.348	0.455	0.485	0.589	0.577
RNN	5	0.575	0.668	**0.773**	0.652	0.597
RNN-LSTM	5	0.644	**0.708**	0.751	**0.810**	**0.876**
RNN-GRU	5	**0.687**	0.749	**0.773**	0.670	0.848

Table notes. The numbers in the table are AUC. Number of features is 5, number of neural units in the hidden layer or RNN hidden state is 10, L2 weight is 0.1, training algorithm is Adam, training time is 100 epochs, minibatch size is 16. All the models use up to five lags of the same variables: real annual house price growth, real annual stock index growth, annual growth in credit-to-GDP ratio, current account-to-GDP ratio, and annual growth in real GDP.

Table 4. Summary of the evaluation results

			Forecast horizon				
Model	Lags	Metric	1	2	3	4	5
		In-sample	0.693	0.710	0.723	0.759	0.618
Logit	1	Validation	0.618	0.655	0.671	0.692	0.420
		Out-of-sample	0.574	0.608	0.713	0.685	0.338
		In-sample	0.831	0.857	0.898	0.897	0.878
Logit	5	Validation	0.649	0.693	0.735	0.755	0.721
		Out-of-sample	0.398	0.366	0.566	0.698	0.736
		In-sample	0.770	0.812	0.788	0.794	0.707
NN	1	Validation	0.608	0.729	0.700	0.687	0.552
		Out-of-sample	0.502	0.551	0.732	0.674	0.479
		In-sample	0.943	0.963	0.973	0.967	0.963
NN	5	Validation	0.687	0.696	0.756	0.682	0.691
		Out-of-sample	0.348	0.455	0.485	0.589	0.577
		In-sample	0.761	0.778	0.813	0.822	0.813
RNN	5	Validation	0.619	0.695	0.726	0.707	0.728
		Out-of-sample	0.575	0.668	**0.773**	0.652	0.597
		In-sample	0.868	0.875	0.933	0.897	0.871
RNN-LSTM	5	Validation	0.761	**0.762**	**0.835**	**0.799**	**0.759**
		Out-of-sample	0.644	**0.708**	0.751	**0.810**	**0.876**
		In-sample	0.870	0.871	0.890	0.898	0.854
RNN-GRU	5	Validation	**0.801**	0.750	0.783	0.795	0.750
		Out-of-sample	**0.687**	0.749	**0.773**	0.670	0.848

Table notes. In-sample refers to using period 1970–2016 for both training and testing. Validation refers to corresponding country-by-country cross-validation (Table 3). Out-of-sample refers to the corresponding sequential evaluation (Table 4c). The numbers in the table are AUC. Number of features is 5, number of neural units in the hidden layer or RNN hidden state is 10, L2 weight is 0.1, training algorithm is Adam, training time is 100 epochs, minibatch size is 16. All the models use up to five lags of the same variables: real annual house price growth, real annual stock index growth, annual growth in credit-to-GDP ratio, current account-to-GDP ratio, and annual growth in real GDP.

Figure 5. Sensitivity of RNN-LSTM prediction performance to training algorithm.

a) A random initialization

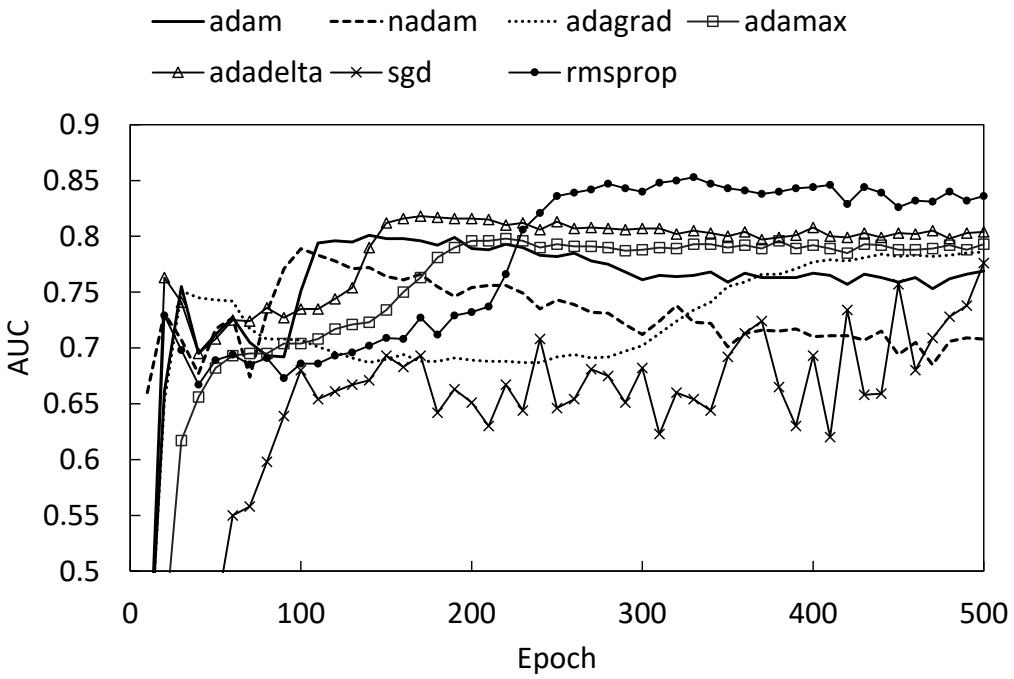


Figure notes. The y-axis is AUC for sequential out-of-sample 3-year ahead prediction. Number of features is 5, number of neural units in the hidden layer or RNN hidden state is 10, L2 weight is 0.1, minibatch size is 16.

b) Another random initialization

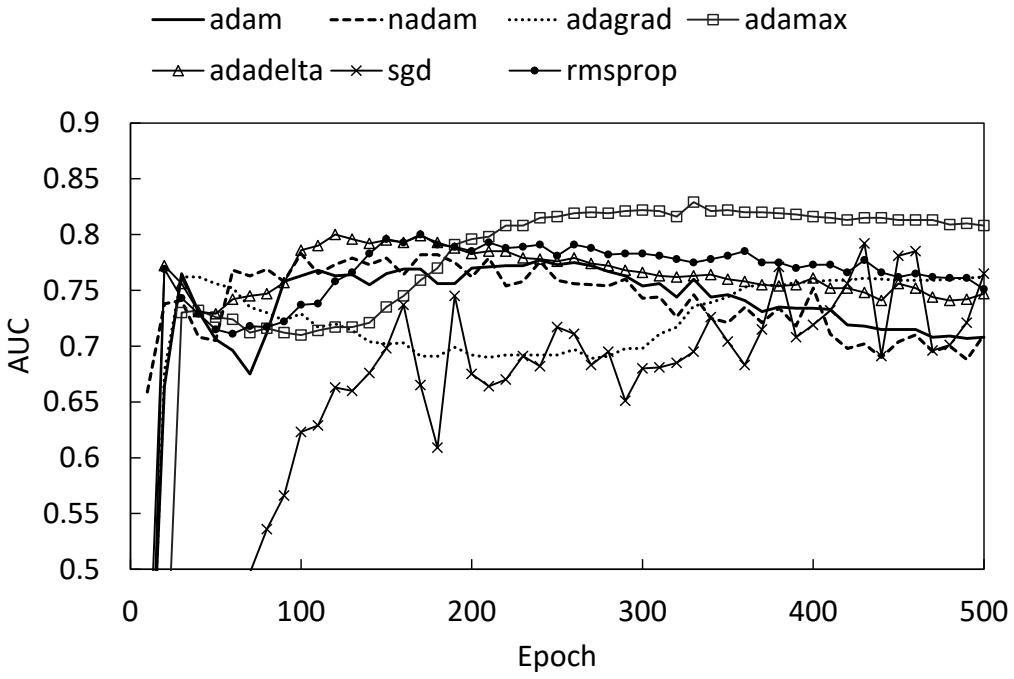


Figure notes. The y-axis is AUC for sequential out-of-sample 3-year ahead prediction. Number of features is 5, number of neural units in the hidden layer or RNN hidden state is 10, L2 weight is 0.1, minibatch size is 16.

Figure 6. Sensitivity of RNN-LSTM prediction performance dropout and L2 regularization.

a) Dropout neurons with probability 0.3

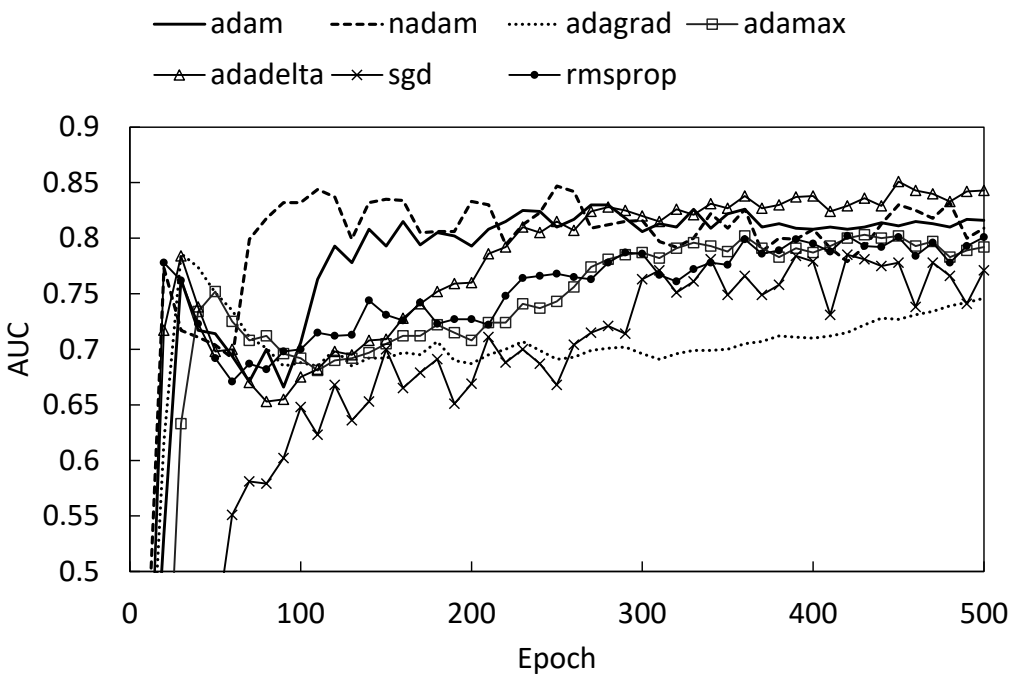


Figure notes. The y-axis is AUC for sequential out-of-sample 3-year ahead prediction. Number of features is 5, number of neural units in the hidden layer or RNN hidden state is 10, L2 weight is 0.1, minibatch size is 16, dropout probability is 0.3.

b) No dropout but increase regularization weight to 0.3

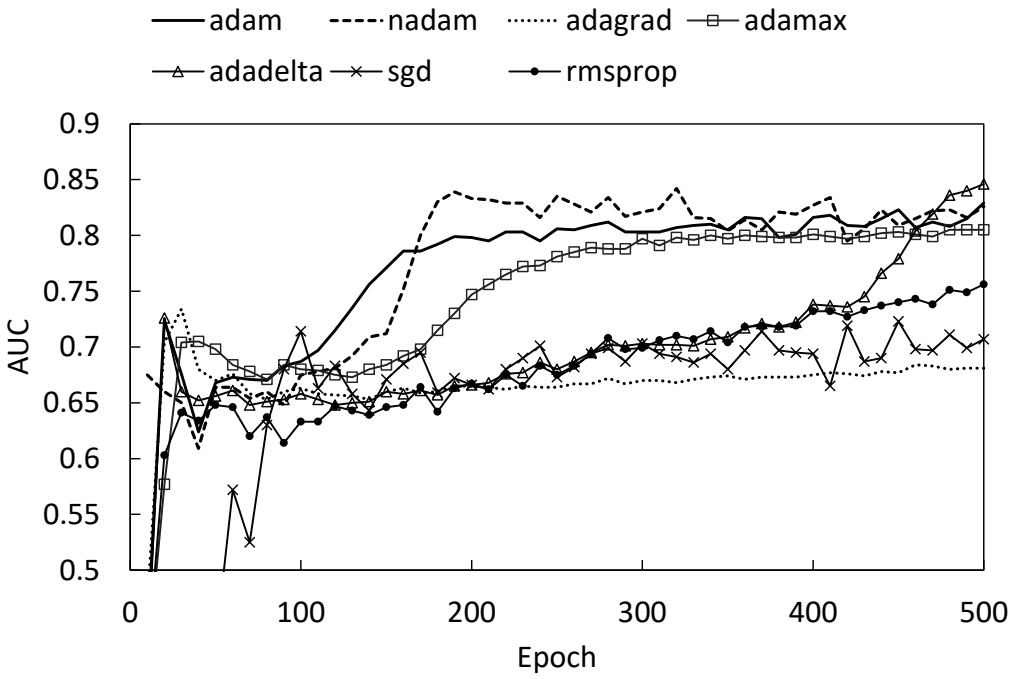


Figure notes. The y-axis is AUC for sequential out-of-sample 3-year ahead prediction. Number of features is 5, number of neural units in the hidden layer or RNN hidden state is 10, L2 weight is 0.3, minibatch size is 16.

Figure 7. Sensitivity of the neural network architectures to L2 regularization

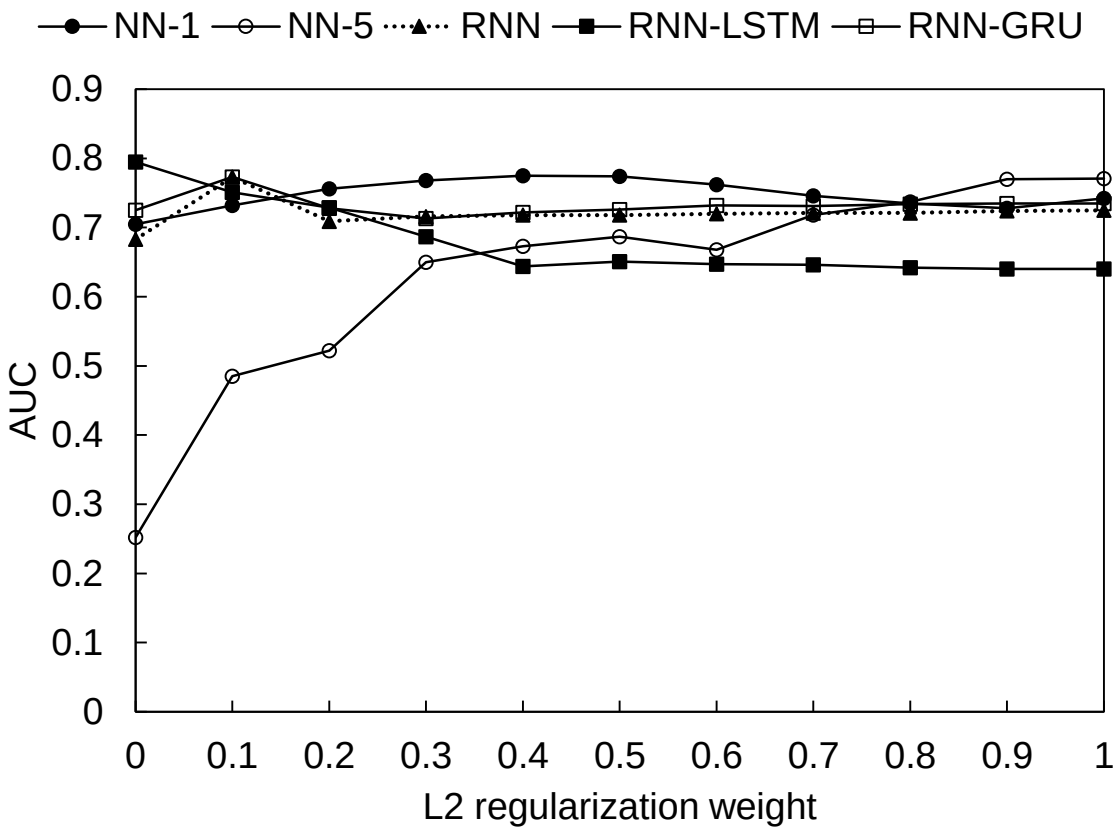


Figure notes. The y-axis is AUC for sequential out-of-sample 3-year ahead prediction. The x-axis is the L2 regularization weight. Number of features is 5, number of neural units in the hidden layer or RNN hidden state is 10, minibatch size is 16.

Figure 8. Sensitivity of RNN-LSTM prediction performance to number of units in the hidden state.

a) Number of features = 5

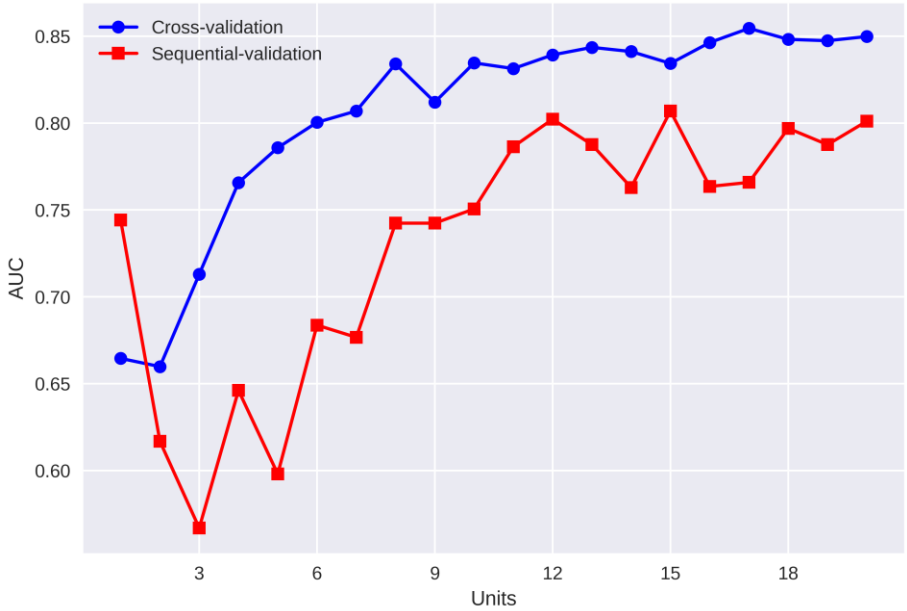


Figure notes. The y-axis is AUC for sequential out-of-sample 3-year ahead prediction. L2 weight is 0.1, training algorithm is Adam, training time is 100 epochs, minibatch size is 16.

b) Number of features = 13

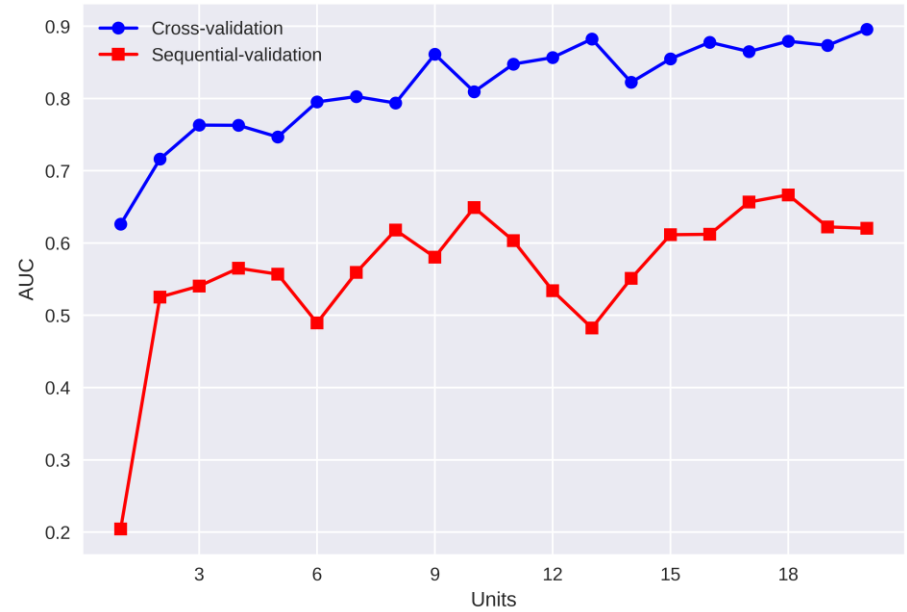


Figure notes. The y-axis is AUC for sequential out-of-sample 3-year ahead prediction. L2 weight is 0.1, training algorithm is Adam, training time is 100 epochs, minibatch size is 16.

Figure 9. Sensitivity of RNN-LSTM prediction performance to number of time steps.

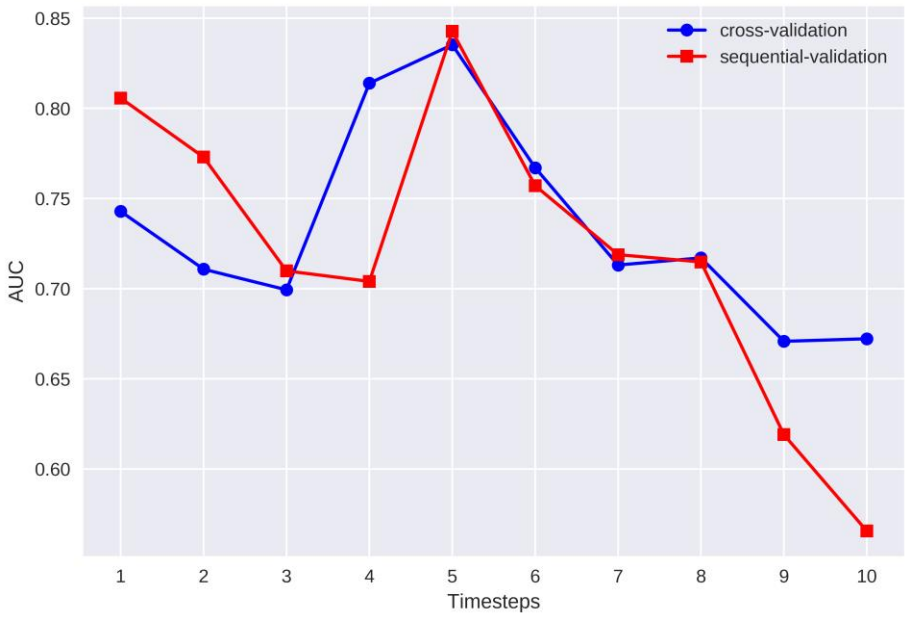


Figure notes. The y-axis is AUC for sequential out-of-sample 3-year ahead prediction. Number of features is 5, number of neural units in the hidden layer or RNN hidden state is 10, L2 weight is 0.1, training algorithm is Adam, training time is 100 epochs, minibatch size is 16.

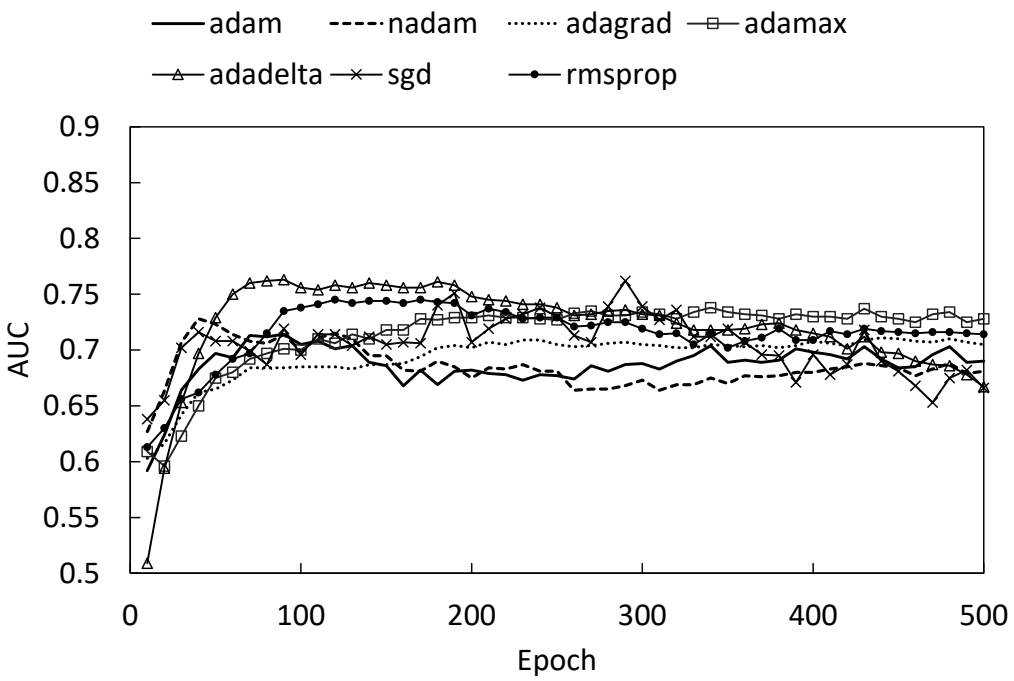
Supplementary material

Figure S1 shows the sensitivity to training algorithm for rest of the neural network architectures. We can see that GRU architecture tends to dominate the simple RNN architecture, which in turn dominates the one hidden layer perceptron architecture. This confirms that the conclusions of the article do not depend on the training algorithm.

Table S1 shows the sequential validation results for the one-hidden layer perceptron when using larger regularization weight. The regularization weight is introduced to reduce overfitting. In panel a) the regularization weight is 0.5, which is optimal for the one-hidden layer perceptron with one lag (based on Figure 7 where the forecast horizon 3 with 1970-2002 training period). In panel b) the regularization weight is 1, which is optimal for the one-hidden layer perceptron with 5 lags (again based on Figure 7 where the forecast horizon 3 with 1970-2002 training period). Fine-tuning the regularization weight improves the performance of the one-hidden layer perceptron. However, the improvement in performance is less stable (i.e. does not generalize well to different forecast horizons or training samples) than for the non-fine-tuned RNN algorithms.

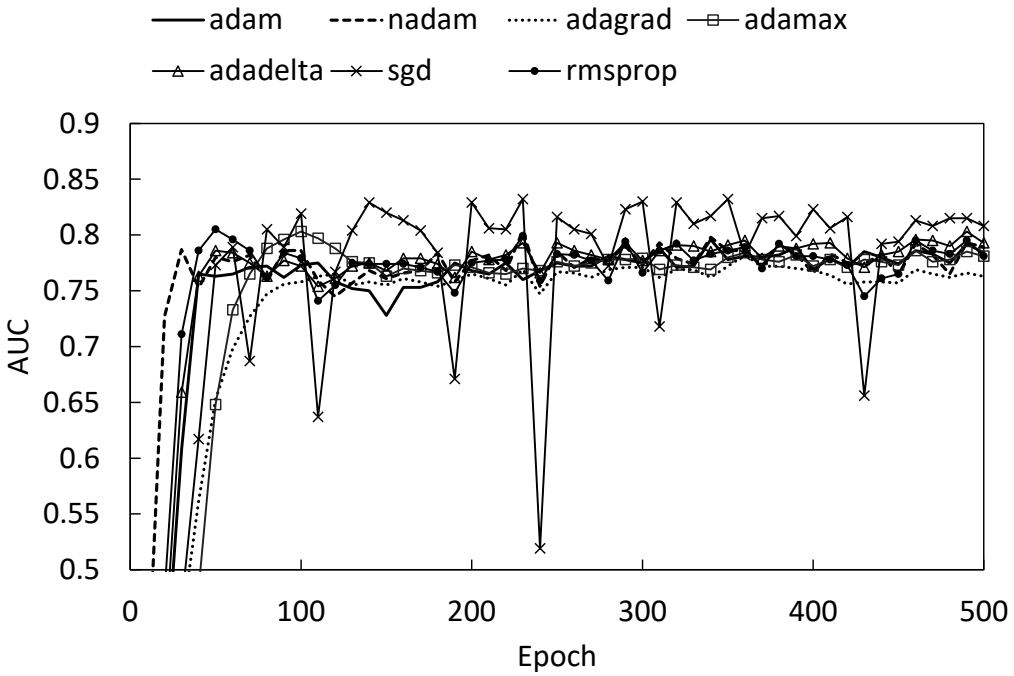
Figure S1. Sensitivity of prediction performance to training algorithm for different network architectures.

a) One hidden layer perceptron (1 lag)



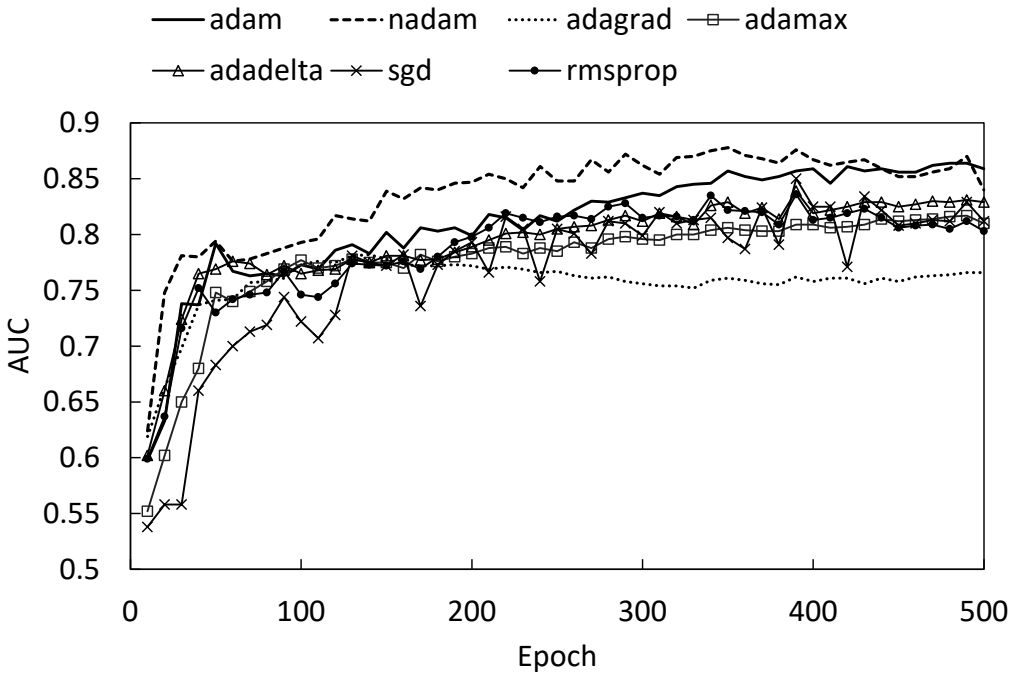
The y-axis is AUC for sequential out-of-sample 3-year ahead prediction. Number of features is 5, number of neural units in the hidden layer or RNN hidden state is 10, L2 weight is 0.1.

b) Recurrent neural network



The y-axis is AUC for sequential out-of-sample 3-year ahead prediction. Number of features is 5, number of neural units in the hidden layer or RNN hidden state is 10, L2 weight is 0.1.

c) GRU neural network



The y-axis is AUC for sequential out-of-sample 3-year ahead prediction. Number of features is 5, number of neural units in the hidden layer or RNN hidden state is 10, L2 weight is 0.1.

Table S1. Sequential-validation results for one hidden-layer perceptron with large regularization weight

The training period is indicated in column 1.

a) L2 regularization weight = 0.5

			Forecast horizon				
Training period	Model	Lags	1	2	3	4	5
1870-2002	NN	1	0.532	0.618	0.777	0.773	0.419
1870-2002	NN	5	0.413	0.482	0.547	0.418	0.749
1946-2002	NN	1	0.517	0.657	0.783	0.734	0.383
1946-2002	NN	5	0.362	0.483	0.489	0.357	0.698
1970-2002	NN	1	0.556	0.651	0.774	0.755	0.470
1970-2002	NN	5	0.444	0.390	0.687	0.804	0.725

Table notes. The numbers in the table are AUC. Number of features is 5, number of neural units in the hidden layer is 10, training algorithm is Adam, training time is 100 epochs, minibatch size is 16. All the models use up to five lags of the same variables: real annual house price growth, real annual stock index growth, annual growth in credit-to-GDP ratio, current account-to-GDP ratio, and annual growth in real GDP.

b) L2 regularization weight = 1

			Forecast horizon				
Training period	Model	Lags	1	2	3	4	5
1870-2002	NN	1	0.508	0.609	0.727	0.782	0.345
1870-2002	NN	5	0.426	0.471	0.592	0.546	0.704
1946-2002	NN	1	0.506	0.646	0.775	0.773	0.308
1946-2002	NN	5	0.406	0.588	0.660	0.560	0.721
1970-2002	NN	1	0.559	0.692	0.742	0.730	0.393
1970-2002	NN	5	0.477	0.453	0.771	0.726	0.823

Table notes. The numbers in the table are AUC. Number of features is 5, number of neural units in the hidden layer is 10, training algorithm is Adam, training time is 100 epochs, minibatch size is 16. All the models use up to five lags of the same variables: real annual house price growth, real annual stock index growth, annual growth in credit-to-GDP ratio, current account-to-GDP ratio, and annual growth in real GDP.

Bank of Finland Research Discussion Papers 2019

ISSN 1456-6184, online

1/2019 Marco Gallegati – Federico Giri – Michele Fratianni
Money growth and inflation: International historical evidence on high inflation episodes for developed countries
ISBN 978-952-323-257-0, online

2/2019 Francesco D'Acunto – Daniel Hoang – Maritta Paloviita – Michael Weber
IQ, Expectations, and Choice
ISBN 978-952-323-258-7, online

3/2019 Orkun Saka
Domestic banks as lightning rods? Home bias and information during Eurozone crisis
ISBN 978-952-323-260-0, online

4/2019 Saara Tuuli
Model-based regulation and firms' access to finance
ISBN 978-952-323-261-7, online

5/2019 Thomas A. Lubik – Christian Matthes – Fabio Verona
Assessing U.S. aggregate fluctuations across time and frequencies
ISBN 978-952-323-263-1, online

6/2019 Karlo Kauko – Eero Tölö
On the long-run calibration of the Credit-to-GDP gap as a banking crisis predictor
ISBN 978-952-323-264-8, online

7/2019 Olli-Matti Laine
The effect of TLTRO-II on bank lending
ISBN 978-952-323-269-3, online

8/2019 Michaela Schmöller
Stagnant wages, sectoral misallocation and slowing productivity growth
ISBN 978-952-323-273-0, online

9/2019 Adam Gulan – Markus Haavio – Juha Kilponen
Can large trade shocks cause crises? The case of the Finnish-Soviet trade collapse
ISBN 978-952-323-274-7, online

10/2019 Ewa Stanisławska – Maritta Paloviita – Tomasz Łyziak
Assessing reliability of aggregated inflation views in the European Commission consumer survey
ISBN 978-952-323-277-8, online

11/2019 Patrick M. Crowley – David Hudgins
U.S. Macroeconomic Policy Evaluation in an Open Economy Context using Wavelet Decomposed Optimal Control Methods
ISBN 978-952-323-279-2, online

12/2019 Luís Aguiar-Conraria – Manuel M. F. Martins – Maria Joana Soares
The Phillips Curve at 60: time for time and frequency
ISBN 978-952-323-280-8, online

13/2019 Gene Ambrocio – Iftekhar Hasan
Friends for the benefits: The effect of political ties on sovereign borrowing conditions
ISBN 978-952-323-282-2, online

14/2019 Eero Tölö
Predicting systemic financial crises with recurrent neural networks
ISBN 978-952-323-287-7, online