Predicting systemic financial crises with recurrent neural networks
Eero Tölö
Predicting systemic financial crises with recurrent neural networks

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

The opinions expressed in this paper are those of the authors and do not necessarily reflect the views of the Bank of Finland.
Predicting systemic financial crises with recurrent neural networks

Eero Tölö*†

21 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 greatly 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 Sciences.

† The work was conducted during spring 2019 at the LSE. 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 predicting major macroeconomic events such as 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. 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) 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, 2016, and Ristolainen 2015, 2017) with some 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, 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 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 Jordà-Schularick-Taylor (JST) macrohistory database, which covers 17 advanced economies for the period 1870-2016.
We train the models to predict financial crises with a prediction horizon extending from 1 to 5 years. Then the models are 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 by a clear margin. 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 predicting recessions, as in Qi (2001), and other types of crises, as in Fioramanti (2008), or more generally in predicting macroeconomic time series (see Cook and Smalter, 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 (2016) benchmark the crisis prediction performance of various machine learning methods including neural networks. Ristolainen (2015, 2018) focuses strictly on the neural network approach and benchmarks against logistic regression only. 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 (2016) 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 (2015, 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.

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 (2016), and Ristolainen (2015, 2017).

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:

\[
    h(X) = a_{\text{relu}}(WX + b) , \tag{1}
\]

\[
    o(h) = a_{\text{sigmoid}}(Vh + c) . \tag{2}
\]

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

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 are considered in the sensitivity analysis.

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, Cook and Hall (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 $h_t$ of dimensionality $h$, which evolves through a number of time steps $T$. The evolution of $h_t$ depends on the previous hidden state $h_{t-1}$ and the current input $X_t$. 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

$$h_t(h_{t-1}, X_t) = \text{tanh}(W X_t + U h_{t-1} + b), \quad t=1,2,...,T,$$ (3)

$$o(h_T) = \text{sigmoid}(V h_T + c).$$ (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 $U$, 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 $h_t$ to $h_{t+1}$ 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 $h_t$ and $s_t$, which both have dimensionality $h$. Gating units $\sigma$ are elementwise sigmoid-functions that control the flow of information at points (x), $x$ denoting Hadamard product $\odot$ (elementwise multiplication).

We denote by $f$, $i$, and $o$ 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:

$$f = \sigma(W x + U h_{t-1} + b),$$ (5)

$$i = \sigma(W x + U h_{t-1} + b),$$ (6)

$$o = \sigma(W x + U h_{t-1} + b),$$ (7)

$$s = f \odot s_{t-1} + i \odot \text{tanh}(W x + U h_{t-1} + b),$$ (8)

$$h = o \odot \text{tanh}(s).$$ (9)
In the main results, the dimensionality of the hidden state is 10, and the number of time steps is 5. Training the LSTM network is similar to training a feedforward neural network as described previously.

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. LSTM cells can complete more complex tasks than GRU cells. However, GRU has been shown to exhibit better performance in some smaller datasets.

Following update equations describe GRU cell:

\[ u_t = \sigma(W_u x_t + U_h h_{t-1} + b_u) \], (10)

\[ r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \], (12)

\[ h_t = u_t \odot h_{t-1} + (1 - u_t) \odot \tanh(W_x x_t + U(r_t \odot h_{t-1}) + b) \]. (13)

In the main results, the dimensionality of the hidden state is 10, and the number of time steps is 5. Training the GRU network is similar to training a feedforward neural network as described previously.

4. Data

All 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. 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 results 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 (2016).

5. Performance evaluation

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 random cross-validation), all the input variables are transformed
stationary, and there is no explicit dependence on time, 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, 1.0, 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. 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. The RNN-LSTM and the RNN-GRU emerge as the most reliable prediction models and consistently outperform the logistic benchmarks as well as the less sophisticated one-hidden-layer perceptron architectures (NN with lags 1 and 1 through 5) and the basic recurrent neural network (RNN). 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 consistently outperforms the logistic benchmarks in all periods albeit with only a narrow margin.

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 for also the other periods. Table 3 presents the AUC evaluated for each model and forecasting horizon. RNN-LSTM and RNN-GRU stand out as the two strongest methods much the same way as before. 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 $T_0 = 1870, 1946, \text{and } 1970$, 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. We can decrease this variability by careful choice of the training parameters as shown in the next section.

**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 focuses on the RNN-LSTM, which performed best in off-the-shelf evaluations presented in the previous section. We first consider sensitivity to training and then sensitivity to complexity of the neural network architecture.

**Sensitivity to training**

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 almost never converge to the same result. Hence, a neural network's prediction performance tends to depend on how we train it. The dependence on the initial values is easy to handle as we can train an ensemble of neural networks and take average prediction. For the training process, besides the algorithm, considerations include different regularization techniques such as L2 regularization and dropout, which are used to reduce overfitting. L2 regularization adds a penalty term proportional to the sum of the squared coefficients. 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 suggests that adding the dropout feature could be helpful and better option than increasing the regularization weight further.

**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 7 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 8 shows the cross-validated prediction performance as a function of time steps.

7. Discussion

Our results can be related 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 (2015, 2017) and Holopainen and Sarlin (2017), we don't find a large performance difference between the logistic regression model and the multilayer perceptron. Two potential reasons are, first, the datasets differ, and second, Ristolainen (2015, 2017) and Holopainen and Sarlin (2016) use higher frequency data and do not include so many lagged values of predictors.

The AUC can’t be compared directly to those in Ristolainen (2015, 2017) and Holopainen and Sarlin (2016) because the datasets are so different. However, compared Fricke (2018) whose dataset is practically equivalent, 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 according to cite ECB).

In future work 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. Also instead of the fixed number of time steps in the RNN, one could try a flexible time step scheme whereby the network input is the full-time series of a country. 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

We have found recurrent neural networks based on LSTM and GRU cells to be efficient tools for predicting systemic financial crises. The capacity to handle time series data seem to be particularly important for the improved performance as compared to single layer perceptron architecture. Overall, the results support the notion that recurrent neural networks could be a useful tool for 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, this may not be true in general. Choosing the optimal parameter values and training the model may require considerable effort.

References

Alessi, Lucia and Antunes, Antonio and Babecky, Jan and Baltussen, Simon and Behn, Markus and Bonfim, Diana and Bush, Oliver and Detken, Carsten and Frost, Jon and Guimaraes, Rodrigo and Havranek, Tomas and Joy, Mark and Kauko, Karlo and Mateju, Jakub and Monteiro, Nuno and Neudorfer, Benjamin and Peltonen, Tuomas A. and Rusnak, Marek and Rodrigues, Paulo Manuel and Schudel, Willem and Sigmund, Michael and Stremmel, Hanno and Smidkova, Katerina and van Tilburg, Ruben and Vasicek, Borek and Zigraiova, Diana, Comparing Different Early Warning Systems: Results from a Horse Race Competition Among Members of the Macro-Prudential Research Network (February 1, 2015). Available at SSRN: https://ssrn.com/abstract=2566165 or http://dx.doi.org/10.2139/ssrn.2566165


Reinhart and Rogoff, 2009, This Time Is Different: Eight Centuries of Financial Folly.


Figure 1. Perceptron with one hidden layer.

Output layer

Hidden layer

Input layer

X1  X2  X3  X4  X5

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.

<table>
<thead>
<tr>
<th>Year</th>
<th>Countries</th>
</tr>
</thead>
<tbody>
<tr>
<td>1870</td>
<td>Belgium, Switzerland</td>
</tr>
<tr>
<td>1871</td>
<td>Japan</td>
</tr>
<tr>
<td>1873</td>
<td>Germany, Italy, USA</td>
</tr>
<tr>
<td>1877</td>
<td>Denmark, Finland</td>
</tr>
<tr>
<td>1878</td>
<td>Sweden</td>
</tr>
<tr>
<td>1882</td>
<td>France</td>
</tr>
<tr>
<td>1883</td>
<td>Spain</td>
</tr>
<tr>
<td>1885</td>
<td>Belgium, Denmark</td>
</tr>
<tr>
<td>1887</td>
<td>Italy</td>
</tr>
<tr>
<td>1889</td>
<td>France</td>
</tr>
<tr>
<td>1890</td>
<td>Japan, Portugal, Spain</td>
</tr>
<tr>
<td>1890</td>
<td>UK</td>
</tr>
<tr>
<td>1891</td>
<td>Germany</td>
</tr>
<tr>
<td>1893</td>
<td>Australia, Italy, Netherlands, USA</td>
</tr>
<tr>
<td>1899</td>
<td>Norway</td>
</tr>
<tr>
<td>1900</td>
<td>Finland</td>
</tr>
<tr>
<td>1901</td>
<td>Germany</td>
</tr>
<tr>
<td>1907</td>
<td>Canada, Germany, Italy, Japan, Netherlands, Sweden, USA</td>
</tr>
<tr>
<td>1908</td>
<td>Denmark</td>
</tr>
<tr>
<td>1910</td>
<td>Switzerland</td>
</tr>
<tr>
<td>1913</td>
<td>Spain</td>
</tr>
<tr>
<td>1920</td>
<td>Japan, Portugal, Spain</td>
</tr>
<tr>
<td>1921</td>
<td>Denmark, Finland, Italy, Netherlands</td>
</tr>
<tr>
<td>1922</td>
<td>Norway, Sweden</td>
</tr>
<tr>
<td>1923</td>
<td>Portugal</td>
</tr>
<tr>
<td>1924</td>
<td>Spain</td>
</tr>
<tr>
<td>1925</td>
<td>Belgium</td>
</tr>
<tr>
<td>1927</td>
<td>Japan</td>
</tr>
<tr>
<td>1929</td>
<td>USA</td>
</tr>
<tr>
<td>1930</td>
<td>France, Italy</td>
</tr>
<tr>
<td>1931</td>
<td>Belgium, Denmark, Finland, Germany, Norway, Portugal, Spain, Sweden, Switzerland</td>
</tr>
<tr>
<td>1934</td>
<td>Belgium</td>
</tr>
<tr>
<td>1935</td>
<td>Italy</td>
</tr>
<tr>
<td>1939</td>
<td>Belgium, Netherlands</td>
</tr>
<tr>
<td>1974</td>
<td>UK</td>
</tr>
<tr>
<td>1977</td>
<td>Spain</td>
</tr>
<tr>
<td>1984</td>
<td>USA</td>
</tr>
<tr>
<td>1987</td>
<td>Denmark</td>
</tr>
<tr>
<td>1988</td>
<td>Norway</td>
</tr>
<tr>
<td>1989</td>
<td>Australia</td>
</tr>
<tr>
<td>1990</td>
<td>Italy</td>
</tr>
<tr>
<td>1991</td>
<td>Finland, Sweden, Switzerland, UK</td>
</tr>
<tr>
<td>1997</td>
<td>Japan</td>
</tr>
<tr>
<td>2007</td>
<td>UK, USA</td>
</tr>
<tr>
<td>2008</td>
<td>Belgium, Denmark, France, Germany, Italy, Netherlands, Portugal, Spain, Sweden, Switzerland</td>
</tr>
</tbody>
</table>
Table 2. Cross-validation results with 1-year forecast horizon.

<table>
<thead>
<tr>
<th>Model</th>
<th>Timestep or lags</th>
<th>Time period</th>
<th>Full sample</th>
<th>Post WW2</th>
<th>1970-2016</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logit</td>
<td>1</td>
<td>0.610</td>
<td>0.623</td>
<td>0.618</td>
<td></td>
</tr>
<tr>
<td>Logit</td>
<td>5</td>
<td>0.637</td>
<td>0.674</td>
<td>0.649</td>
<td></td>
</tr>
<tr>
<td>NN</td>
<td>1</td>
<td>0.596</td>
<td>0.606</td>
<td>0.569</td>
<td></td>
</tr>
<tr>
<td>NN</td>
<td>5</td>
<td>0.652</td>
<td>0.680</td>
<td>0.655</td>
<td></td>
</tr>
<tr>
<td>RNN</td>
<td>5</td>
<td>0.654</td>
<td>0.692</td>
<td>0.619</td>
<td></td>
</tr>
<tr>
<td>RNN-LSTM</td>
<td>5</td>
<td>0.714</td>
<td>0.774</td>
<td>0.761</td>
<td></td>
</tr>
<tr>
<td>RNN-GRU</td>
<td>5</td>
<td>0.762</td>
<td>0.798</td>
<td>0.801</td>
<td></td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Model</th>
<th>Forecast horizon</th>
<th>Timestep or lags</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logit</td>
<td>1</td>
<td>0.618</td>
<td>0.655</td>
<td>0.671</td>
<td>0.692</td>
<td>0.420</td>
<td></td>
</tr>
<tr>
<td>Logit</td>
<td>5</td>
<td>0.649</td>
<td>0.693</td>
<td>0.735</td>
<td>0.755</td>
<td>0.721</td>
<td></td>
</tr>
<tr>
<td>NN</td>
<td>1</td>
<td>0.569</td>
<td>0.692</td>
<td>0.682</td>
<td>0.598</td>
<td>0.515</td>
<td></td>
</tr>
<tr>
<td>NN</td>
<td>5</td>
<td>0.655</td>
<td>0.676</td>
<td>0.674</td>
<td>0.651</td>
<td>0.632</td>
<td></td>
</tr>
<tr>
<td>RNN</td>
<td>5</td>
<td>0.619</td>
<td>0.695</td>
<td>0.726</td>
<td>0.707</td>
<td>0.728</td>
<td></td>
</tr>
<tr>
<td>RNN-LSTM</td>
<td>5</td>
<td>0.761</td>
<td><strong>0.762</strong></td>
<td><strong>0.835</strong></td>
<td><strong>0.799</strong></td>
<td><strong>0.759</strong></td>
<td></td>
</tr>
<tr>
<td>RNN-GRU</td>
<td>5</td>
<td><strong>0.801</strong></td>
<td>0.750</td>
<td>0.783</td>
<td>0.795</td>
<td>0.750</td>
<td></td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Model</th>
<th>Forecast horizon</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logit</td>
<td>1</td>
<td>0.535</td>
<td>0.616</td>
<td>0.744</td>
<td>0.757</td>
<td>0.417</td>
</tr>
<tr>
<td>Logit</td>
<td>5</td>
<td>0.505</td>
<td>0.521</td>
<td>0.671</td>
<td>0.671</td>
<td>0.633</td>
</tr>
<tr>
<td>NN</td>
<td>1</td>
<td>0.370</td>
<td>0.521</td>
<td>0.753</td>
<td>0.599</td>
<td>0.567</td>
</tr>
<tr>
<td>NN</td>
<td>5</td>
<td>0.336</td>
<td>0.314</td>
<td>0.286</td>
<td>0.535</td>
<td>0.609</td>
</tr>
<tr>
<td>RNN</td>
<td>5</td>
<td>0.557</td>
<td>0.616</td>
<td>0.781</td>
<td>0.701</td>
<td>0.656</td>
</tr>
<tr>
<td>RNN-LSTM</td>
<td>5</td>
<td>0.584</td>
<td>0.742</td>
<td>0.643</td>
<td>0.731</td>
<td>0.850</td>
</tr>
<tr>
<td>RNN-GRU</td>
<td>5</td>
<td>0.531</td>
<td>0.721</td>
<td>0.728</td>
<td>0.745</td>
<td>0.820</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Model</th>
<th>Forecast horizon</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logit</td>
<td>1</td>
<td>0.558</td>
<td>0.602</td>
<td>0.748</td>
<td>0.672</td>
<td>0.458</td>
</tr>
<tr>
<td>Logit</td>
<td>5</td>
<td>0.426</td>
<td>0.41</td>
<td>0.513</td>
<td>0.575</td>
<td>0.622</td>
</tr>
<tr>
<td>NN</td>
<td>1</td>
<td>0.371</td>
<td>0.567</td>
<td>0.763</td>
<td>0.612</td>
<td>0.594</td>
</tr>
<tr>
<td>NN</td>
<td>5</td>
<td>0.435</td>
<td>0.367</td>
<td>0.196</td>
<td>0.433</td>
<td>0.632</td>
</tr>
<tr>
<td>RNN</td>
<td>5</td>
<td>0.551</td>
<td>0.641</td>
<td>0.799</td>
<td>0.685</td>
<td>0.694</td>
</tr>
<tr>
<td>RNN-LSTM</td>
<td>5</td>
<td>0.595</td>
<td>0.717</td>
<td>0.735</td>
<td>0.786</td>
<td>0.873</td>
</tr>
<tr>
<td>RNN-GRU</td>
<td>5</td>
<td>0.609</td>
<td>0.744</td>
<td>0.727</td>
<td>0.678</td>
<td>0.838</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Model</th>
<th>Forecast horizon</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Timestep or lags</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logit</td>
<td>1</td>
<td>0.574</td>
<td>0.608</td>
<td>0.713</td>
<td>0.685</td>
<td>0.338</td>
</tr>
<tr>
<td>Logit</td>
<td>5</td>
<td>0.398</td>
<td>0.366</td>
<td>0.566</td>
<td>0.698</td>
<td>0.736</td>
</tr>
<tr>
<td>NN</td>
<td>1</td>
<td>0.431</td>
<td>0.574</td>
<td>0.705</td>
<td>0.596</td>
<td>0.596</td>
</tr>
<tr>
<td>NN</td>
<td>5</td>
<td>0.432</td>
<td>0.373</td>
<td>0.252</td>
<td>0.540</td>
<td>0.692</td>
</tr>
<tr>
<td>RNN</td>
<td>5</td>
<td>0.575</td>
<td>0.668</td>
<td>0.773</td>
<td>0.652</td>
<td>0.597</td>
</tr>
<tr>
<td>RNN-LSTM</td>
<td>5</td>
<td>0.644</td>
<td>0.708</td>
<td>0.751</td>
<td>0.810</td>
<td>0.876</td>
</tr>
<tr>
<td>RNN-GRU</td>
<td>5</td>
<td>0.687</td>
<td>0.749</td>
<td>0.773</td>
<td>0.670</td>
<td>0.848</td>
</tr>
</tbody>
</table>

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

a) A random initialization.

![Graph showing sensitivity of RNN-LSTM prediction performance to training algorithm.]

Figure notes. The figure is for sequential evaluation. 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, minibatch size is 16.

b) Another random initialization.

![Graph showing sensitivity of RNN-LSTM prediction performance to training algorithm.]

Figure notes. The figure is for sequential evaluation. 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, minibatch size is 16.
Figure 6. Effect of dropout and stronger regularization.

a) Dropout neurons with probability 0.3.

![Graph showing the effect of dropout on AUC](image)

Figure notes. The figure is for sequential evaluation. 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, minibatch size is 16, dropout probability is 0.3.

b) No dropout but increase regularization weight to 0.3.

![Graph showing the effect of increase regularization on AUC](image)

Figure notes. The figure is for sequential evaluation. 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 0.3, minibatch size is 16.
Figure 7. Sensitivity of RNN-LSTM prediction performance to number of units in the hidden state.

a) Number of features = 5.

Figure notes. The numbers in the table are AUC. 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 numbers in the table are AUC. L2 weight is 0.1, training algorithm is Adam, training time is 100 epochs, minibatch size is 16.
Figure 8. Sensitivity of RNN-LSTM prediction performance to number of time steps.

Figure 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.
Annex - Additional sensitivity analysis

Figure A1 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 further confirms the observations in the main article.

Figure A1. Sensitivity of prediction performance to training algorithm for different network architectures.

a) One hidden layer perceptron.

b) Recurrent neural network.
c) GRU neural network.
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 – Iftikhar 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