GDP Forecasting using Recurrent Neural Networks

Cameron Fen and Samir Undavia*

December 21, 2020

Please do not cite or circulate without permission

We study the problem of GDP forecasting, introducing a neural network model that consistently outperforms state-of-the-art economic models. The model outperforms an autoregressive model with 2 lags (AR(2)), and a dynamic stochastic general equilibrium (DSGE) model over all horizons, a factor model on horizons longer than 2 periods ahead, and the median forecast of the Survey of Professional Forecasters at 5 quarters ahead. Forecasts over different time windows, model specifications, along with Monte Carlo simulations suggest the performance of our model is robust and reproducible. Additionally, our tests show performance does not depend significantly on the initialization of parameters, reasonable changes in architecture, and noise due to stochastic optimization. Forecasts evaluation across countries also suggests this model is able to successfully forecast GDP across many different policy regimes, jointly.

*Corresponding author Cameron Fen is PhD student at the University of Michigan, Ann Arbor, MI, 48104 (E-mail: camfen@umich.edu. Website: cameronfen.github.io.). Samir Undavia is an ML and NLP engineer (E-mail: samir@undavia.com). The authors thank Xavier Martin Bautista, Thorsten Drautzburg, Florian Gunsilius, Jeffrey Lin, Daniil Manaenkov, and Matthew Shapiro for helpful discussion and/or comments. We would like to thank AI Capital Management for computation resources. All errors are our own.

I. Introduction

Times series forecasting is an essential part of economics. Since datasets often are small, especially in macroeconomics, this often precludes using all but the simplest of linear models. The lack of variation in linear models makes new innovations difficult as one can make few improvements before the model become to complex to fit the data well. One widely used class of models is autoregressive models, which were introduced in the 1930s (Walker, 1931) and are still used today. In this paper, we seek to improve forecasting models by sidestepping these constraints, which allows us to introduce a neural network model trained on a cross section of 50 countries. Our main result is that this model outperform benchmark economic models like the AR(2), Smets-Wouters DSGE (Smets and Wouters, 2007), and a factor model (Stock and Watson, 2002*a*), (Stock and Watson, 2002*b*), when forecasting GDP.

A central responsibility for econometricians is to predict macroeconomic outcomes. Even being able to accurately forecast GDP is an important step in informing central bankers and treasury officials regarding monetary and fiscal policy. Additionally, our training procedure and the use of neural networks can be applied to forecasting other macroeconomic variables as well.

Our model outperforms an AR(2) and a DSGE model over all horizons, and a factor model on horizons longer than 2 periods ahead. However while our model uses a cross section of countries' data, we only use three covariates, GDP, consumption, and unemployment lags. The DSGE model uses 11 different covariates and the factor model uses 248, making both more difficult to apply in countries where data is sparse. Over the longer horizons (3-5 quarters ahead) our model approaches Survey of Professional Forecasters' (None, 1968) median forecast performance, albeit evaluated on 2020 vintage data (see Appendix G.), culminating in outperformance over that benchmark at 5 quarters ahead. As

the SPF is an ensemble of both models and human decision makers and many recessions like the recent COVID recession aren't picked up by models, we think this performance is noteworthy.

We test across different time periods and model architectures, and for each test, we estimate the model 100 times. We report the test performance of the best model (on a validation set), the performance of the median forecast, the mean forecast, and mean performance of all the models. The outperformance across our main results and robustness tests suggest that the model is numerically stable, robust to specification differences and can be reproduced with wide variety of discoverable hyper-parameter/architecture choices.

Another contribution of this paper is to provide support for cross-sectional country data to discipline larger models. Forecasters building predictive models often suffer from a lack of data. This problem is pronounced in macroeconomics as there has only been around 80 years of macroeconomic data that is high enough in quality for general use. Our paper builds upon the literature that studies the use of cross sectional data to improve counterfactual analysis (Miyamoto, Nguyen and Sergeyev, 2018) (Crandall, Lehr and Litan, 2007) or have a more micro-economic forecasting focus (Baltagi, 2008). In contrast to these literatures, our approach attempts to use a cross section of 50 countries to improve general purpose macroeconomic forecasting. Our method requires a larger data set and a flexible model that can take advantage of these data. We conjecture that such approaches could be successful when applied to non-neural network models, as long as these models are flexible enough.

Our third contribution is to illustrate that reduce form models, particularly our neural network model, are able to generalize across policy regimes when they are estimated on a large cross section of data. This supports the argument that more data, particularly across different regimes, is a more important factor

in overcoming the Lucas critique than choosing between a reduced form versus a structural model. Based on natural language processing studies like Kaplan et al. (2020), we hypothesize that in order to fully take advantage of the cross sectional data, one needs to estimate larger models like neural networks or large-scale DSGE models.

A central reason for the outperformance of our neural network, even when using our data augmentation scheme on baseline models, is because of the long term memory of our recurrent neural network. Linear state space and related models perform poorly on forecasting tasks because the inductive bias is for the models state to return to a steady state value (Panzner and Cimiano, 2016). The consequence of this is the state can only capture recent information as on an intuitive level, once it has returned to a steady state, the model contains no information on previous inputs. However, recurrent neural networks are constructed with gates and so the states can behave periodically or even chaotically without ever returning to a steady state or becoming unbounded (Zerroug, Terrissa and Faure, 2013). Because of this, studies have demonstrated that these models can remember and act on information that was even 100 time steps in the past (Chung et al., 2014).

The paper proceeds as follows: Section II. reviews the literature on forecasting and recurrent neural networks and describes how our paper merges these two fields; Section III. discusses feed-forward neural networks, linear state space models, and gated recurrent units (Cho et al., 2014); Section IV. describes the data; Section V. briefly mentions our model architecture; Section VI. discusses the benchmark models and the SPF that we compare our model to; Sections VII. and VIII. provide the main results and robustness checks; Section IX. concludes the paper.

II. Literature Review

This paper connects two strands of literature: time-series econometrics and deep learning:

In deep learning, recurrent neural networks have been around in many forms, but were mainly popularized in the machine learning literature in the 1980s (Rumelhart, Hinton and Williams, 1986). The popularity and performance of recurrent neural networks grew with the introduction of long short term memory (LSTM) networks with Hochreiter and Schmidhuber (1997). The model uses gates to prevent exploding or disappearing gradients allowing this model to have long term memory. In addition to its pervasive use in language modeling, long short term memory networks are used in fields as disparate as robot control (Mayer et al., 2006), protein homology detection (Hochreiter, Heusel and Obermayer, 2007), and rainfall detection (Shi et al., 2015). We use a modification of long short term memory networks called a gated recurrent unit (Cho et al., 2014). Within economics, gated recurrent units (GRUs) have been applied in stock market prediction (Minh et al., 2018) and power grid load forecasting (Li, Zhuang and Zhang, 2020).

Autoregressive models are the workhorse forecasting models since the late 1930s (see Diebold (1998) and Walker (1931)). Despite its simplicity and age, the model is still used among practitioners and as a baseline in many papers (Watson, 2001). One advancement in forecasting stems from the greater adoption of structural or pseudo-structural time series models like the Smets-Wouters DSGE models (Smets and Wouters, 2007). While DSGE forecasting is widely used in the literature, they are competitive with, but often times no better than, a simple AR(2), with more bespoke DSGE models performing poorer (Edge, Kiley and Laforte, 2010). However, the use of DSGE models for counterfactual analysis is an important and unique benefit of these models. The final model

is the factor model (Stock and Watson, 2002a). The factor model attempts to use a large cross section of data which results in a more comprehensive picture of the economy to perform forecasting. Details on all these models and our implementation can be found in Appendix H. In addition, our paper uses tools from forecast evaluation (West, 1996), (Pesaran and Timmermann, 1992) and (Diebold and Mariano, 2002), as well as model averaging (Leamer and Leamer, 1978), (Koop, Leon-Gonzalez and Strachan, 2012).

III. Neural Network Models

III.A. Feed-Forward Neural Networks

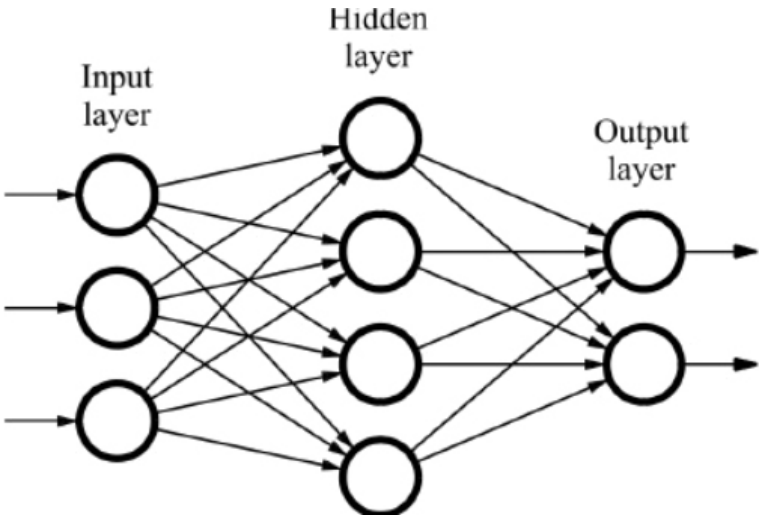

Figure I: An Example of a Feed-Forward Neural Network

The feed-forward network is the prototypical image associated with deep learning. At its core, a feed-forward neural network is a recursively nested linear regression with non-linear transforms. For example, assume X is a vector valued input to the neural network and X_{out} is the output. In a typical linear regression, $X_{out} = X_{in}\beta_1$. The insight for composing a feed-forward network is to take the

output and feed that into another linear regression: $Y = X_{out}\beta_2$. In figure I, X_{in} would be the input layer, X_{out} would be the hidden layer and Y would be the output layer. The problem is not all that interesting if X_{out} is a scalar. If X_{in} is a matrix of dimension: time steps by regressors, X_{out} can be a matrix of dimension time steps by hidden units. Here in the figure, the dimension of the hidden layer is 4. Thus, β_1 has to be a matrix of dimension 3 by 4 (regressors by hidden units). Thus, we make X_{out} an input into multidimensional regression for the second layer, $Y = X_{out}\beta_2$, if the first layer is a vector regression[1]. This can be repeated for as many layers as desired.

Now a composition will result in: $Y = X_{out}\beta_2 = (X_{in}\beta_1)\beta_2$. A product of two matrices is still another matrix, which makes this model still linear regardless of how many layers there are. However, an early result in the literature showed that if between every regression, eg $X_{out} = X_{in}\beta_1$, one inserts an almost arbitrary non-linear link function this allows a neural network to approximate any continuous function (Hornik, Stinchcombe and White, 1989). For example, inserting a logistic transformation between X_{in} and X_{out} i.e. $X_{out} = \sigma(X_{in}\beta_1)$ where $\sigma(z) = \frac{1}{1+e^{-z}}$. One can put these non-linearities as often as one would like to get something like this: $Y = \sigma(\sigma(X_{in}\beta_{1,1})\beta_2)$. These are the fundamental building blocks of neural networks and additional composition of link functions and matrix multiplication of parameters form the basis of deeper networks and allows these models to be universal approximators.

III.B. The Simplest Recurrent Neural Network: A Linear State Space Model

For the purposes of this paper, conceptualizing a recurrent neural network as a distinct model from feed-forward networks will not hinder comprehension.

1. Note: this regression is not a vector autoregression as X_{out} is a latent variable

That being said, without even knowing it, many economists are already familiar with recurrent neural networks. The simplest is a Kalman filter-like linear state space model. The two equations that define the linear state space model are[2]:

$$\mathbf{s}_t = \mathbf{s}_{t-1}\mathbf{U}^s + \mathbf{y}_{t-1}\mathbf{W}^s + \mathbf{b}^s, \tag{1}$$

$$\mathbf{y}_t = \mathbf{s}_t\mathbf{U}^y + \mathbf{y}_{t-1}\mathbf{W}^y + \mathbf{b}^y \tag{2}$$

In a linear state space model, the state $\mathbf{s}_i$ is an unobserved variable which allows the model to keep track of the current environment. One uses the state, along with lagged values of the observed variables, to forecast observed variables $\mathbf{y}_i$. For example, for GDP, the state can be either a expansionary period or a recession. A priori, the econometrician does not know. However, one can make an educated guess based on GDP growth. As machine learning is more interested in prediction, the state is estimated with point estimates, which allows the data scientist to sidestep the tricky problem of filtering. A more detailed explanation of this estimation process is discussed in Appendix C.

The problem with linear state space models is that if one does not apply filtering, the state vector either blows up or goes to a steady state value. This can be seen by recognizing that each additional time step results in the state vector getting multiplied by $\mathbf{U}^s$ an additional time. Depending on if the eigenvectors of $\mathbf{U}^s$ are greater than or less than one, the states will ultimately explode or go to a steady state. More sophisticated neural networks like gated recurrent units (Cho et al., 2014) we use, fix this with the use of gates.

III.C. Gated Recurrent Units

Gated recurrent units (Cho et al., 2014) were introduced to improve upon the performance over recurrent neural networks that resembled linear state space

2. We add autoregressive lags to make the model more general.

models and can deal with this exploding gradient problem discussed above. First we redefine sigma as the logistic link function:

$$\sigma(x) = \frac{e^{\beta x}}{1 + e^{\beta x}} \tag{3}$$

The idea behind the gate, is to allow the model to control the magnitude of the state vector. A simple gated recurrent neural network looks like the linear state space model with an added gate equation:

$$\mathbf{y}_t = \mathbf{h}_t \mathbf{U}^y + E * \mathbf{y}_{t-1} \mathbf{W}^y + \mathbf{b}^y \tag{4}$$

$$\mathbf{z}_t = \sigma(\mathbf{h}_{t-1} \mathbf{U}^h + \mathbf{y}_{t-1} \mathbf{W}^h + \mathbf{b}^h) \tag{5}$$

$$\mathbf{s}_t = \mathbf{h}_{t-1} \mathbf{U}^s + \mathbf{y}_{t-1} \mathbf{W}^s + \mathbf{b}^s \tag{6}$$

$$\mathbf{h}_t = \mathbf{z}_t \odot \mathbf{s}_t \tag{7}$$

The output of $\sigma()$ is a number between zero and one which is element-wise multiplied by $\mathbf{s}_t$ which is the first draft of the state. The operation $\odot$ indicates element-wise multiplication or the Hadamard product. Variables are subscripted with the time period they are observed in (t or $t-1$). Weight matrices, which are not a function of the inputs, are super-scripted with the equation name they feed into. All elements are considered vectors and matrices, and matrix multiplication is implied when no operation is present.

The presence of the gate controls the behavior of the state, which means that even if the eigenvalues of $\mathbf{U}^s$ were greater than one, or equivalently even if $\mathbf{h}_t$ would explode without the gate, the gate can keep the state bounded. Additionally, the steady state distribution of the state does not have to converge to a number. The behavior could be periodic, or even chaotic (Zerroug, Terrissa and Faure, 2013). This allows for the modeling of more complex behavior as

well as the ability of the state vector to "remember" behavior over longer time periods (Chung et al., 2014).

The equations of the gated recurrent unit are:

$$\mathbf{y}_t = \mathbf{h}_t \mathbf{U}^y + E * \mathbf{y}_{t-1} \mathbf{W}^y + \mathbf{b}^y \tag{8}$$

$$\mathbf{z}_t = \sigma(\mathbf{x}_t \mathbf{U}^z + \mathbf{h}_{t-1} \mathbf{W}^z) \tag{9}$$

$$\mathbf{r}_t = \sigma(\mathbf{x}_t \mathbf{U}^r + \mathbf{h}_{t-1} \mathbf{W}^r) \tag{10}$$

$$\mathbf{s}_t = \tanh(\mathbf{x}_t \mathbf{U}^s + (\mathbf{h}_{t-1} \odot \mathbf{r}_t) \mathbf{W}^s) \tag{11}$$

$$\mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{s}_t + \mathbf{z}_t \odot \mathbf{h}_{t-1} \tag{12}$$

Where tanh is defined as the hyperbolic tangent:

$$\tanh(x) = \frac{e^{2*x} - 1}{e^{2*x} + 1} \tag{13}$$

Like the linear state space model, the state vector of the gated recurrent unit persists over time steps in the model. Mapping these equations to equation 4 through 7, equation 8 is the measurement equation (analogous to equation 4.).Equation 9 and 10 are both gates and analogous to equation 5. Equation 11 is the first draft of the state before the gate $\mathbf{z}_t$ is applied and resembles equation 6. Equation 12 is the final draft of the state after $\mathbf{z}_t$ is applied and resembles equation 7. The equations of this model are discussed more in Appendix D.

The recurrent neural network is optimized via gradient descent, where the derivative of the loss function with respect to the parameters is calculated via the chain rule/reverse mode differentiation. The gradient descent optimizer algorithm we use is Adam (Kingma and Ba, 2014), which shares similarities with a quasi-Newton approach. See Appendix I for more information.

IV. Data

United States macroeconomic data is not sufficient enough to train a complex model like a neural network. To fix this, we use data from 49 other developed and developing countries as listed in Appendix A. We source cross country data from Trading Economics via the Quandl API[3] as well as GDP data from the World Bank[4]. We use GDP, consumption, and the unemployment rate as inputs to the model. GDP and consumption are all expressed in growth rates. Unemployment is expressed as a rate as well.

We split our data into training, validation, and test sets. The test set is from 2008-Q4 to 2020-Q1. The validation consists of data from 2003-Q4 to 2008-Q3. The training set is all periods before 2003-Q4 depending on when each country started releasing their data. We chose these periods so that both the test set and the validation set would have both periods of expansion and recession, based on the US business cycle. Including the 2001 recession in the validation set would leave the model without enough training data, so we split the data of the Great Recession over the test and validation set. The quarter with the fall of Lehman Brothers and the largest dip in GDP was the first quarter in our test set, 2008-Q4. There are two quarters with negative growth preceding this which are in the validation set.

V. Model Architecture

The model architecture involves "pre-processing" the data using two feed-forward dense layers with 64 and 128 hidden units respectively and rectified linear unit (Agarap, 2018) activation (see Appendix E.), then running a gated

3. https://www.quandl.com/tools/api
4. World Development Indicators, The World Bank

recurrent unit on this "pre-processed" data[5].

The model design involves the use of batch normalization (Ioffe and Szegedy, 2015) and skip connections (He et al., 2015) between each layer, along with gated recurrent unit layers and dense (normal neural network) layers interspersed. More detail can be found in Appendix F. These design choices all improved forecasting performance. We also use batch normalization (Ioffe and Szegedy, 2015). More details on batch normalization can also be found in Appendix F.

We estimated the model with squared loss and evaluated it with root mean squared error. US forecast only use US validation data, while global forecasts use global validation data. The final test loss evaluates the performance of the model at the given horizon on US GDP data, except in the case where global forecasts are being evaluated. This makes our forecast errors comparable to the SPF and other US macroeconomic forecasts.

VI. Economic Models

The recurrent neural network is compared against a variety of workhorse economic forecasting models. Additionally, we provide comparisons to the Survey of Professional Forecasters (None, 1968) median GDP forecast, which is seen as a proxy for state-of-the-art performance. A discussion of the Survey of Professional Forecasters and our attempt to evaluate their forecasts is contained in Appendix G. The baseline economic models we used are the AR(2) autoregressive model, the Smets-Wouters 2007 DSGE model (Smets and Wouters, 2007), and a factor model (Stock and Watson, 2002*a*) and (Stock and Watson, 2002*b*) and a VAR(1) (Sims, 1980). For the reduced form models, getting cross country data is straightforward, thus we compare those models estimated only on US

5. While we use the word pre-processed, the approach is trained entirely end-to-end and not a two step process as the word "pre-process" might imply

data as well as on our data set of 50 countries. A more detailed explanation of these models is contained in Appendix H.

As is standard with economic forecasting, the baseline models are trained in a pseudo-out-of-sample fashion where the training set/validation set expands as the forecast date becomes more recent. However, with our neural network, we keep the training set and validation set fixed due to computational costs. We suspect our model will likely improve if we used a pseudo-out-of-sample approach.

VII. Results

Table I shows the root mean squared error performance of our gated recurrent unit model in comparison to all other models. We compared the best performing neural network model on the validation data set, the mean neural network prediction over 100 models, and the median over 100 runs with the three baseline models discussed in the economic models section. The time horizon involved for test evaluation is discussed in the data section. We bold any neural network model that outperforms the best economic model. If none of our neural networks outperform this benchmark, we bold the best performing benchmark model. Our recurrent neural network model outperforms all benchmark models for forecasts over 2 quarters ahead. Likewise, we outperform all benchmark models except for the factor model over 1 and 2 quarters ahead. However, compared to the factor model, our neural network only uses GDP, consumption and unemployment rate.

Our neural network model is also competitive with the median forecast of the SPF over longer horizons. In fact, in our Monte Carlo tests (table II) and our 5 period ahead forecasts in table I, all neural network RMSE metrics outperform the SPF, which we denote with a *. In practice, the SPF uses the

TABLE I: GRU Root Mean Squared Error

The performance of the best, mean, and median forecasts for the performance of gated recurrent units on the test set 2009-Q1 to 2020-Q1 (lower is better)

Time (Q's Ahead)	1Q	2Q	3Q	4Q	5Q
VAR(1)					
US Data	2.9	3.8	4.4	4.5	4.7
World Data	2.5	2.9	3.0	3.1	3.1
AR(2)					
US Data	2.5	2.9	3.0	3.1	3.1
World Data	2.6	2.6	2.7	2.7	2.7
Smets Wouters DSGE					
US Data	2.8	3.0	2.9	2.8	2.7
Factor					
US Data	**2.2**	**2.5**	2.5	2.7	2.9
GRU*					
Best	2.4	2.5	2.6	**2.6**	**2.6***
Mean Forecast	2.3	2.5	**2.5**	**2.6**	**2.6***
Median Forecast	2.3	2.5	**2.5**	**2.6**	**2.6***
SPF Median	1.9	2.1	2.4	2.5	2.7

*All GRU models use entire world data cross section

real time vintage data for forecasting, but we use data that is current to 2020 (see Appendix G.). That being said, even with this caveat, a model approaching SPF performance is noteworthy.

Another insight to highlight is that using cross country data to augment US time series improves the performance of reduced form models. In both the VAR(1) and AR(2) case, US forecast performance increased when using the global cross section to estimate the model. In the VAR(1) case, this significantly improved an ineffectual forecasting model to become competitive with the other baseline models. However, neither model trained on world data outperforms our gated recurrent unit model. Despite this, the improvement in all the reduced form models when given a cross section of macro data that they can be improved by adding slightly off task data. This also ends credence to the idea that having a wide range of data that spans policy regimes is more important to building policy invariant models, rather than a distinction between structural and reduced form.

VIII. Robustness Checks

In order to analyze the strengths and weakness of the model, we display the forecasts of our neural network model (median forecast of the 100 models) and compare it to the true forecast along with a selection of baseline models and the SPF in a graph. Below is a picture of the performance of these models along the 3 quarters ahead horizon when the neural network starts to outperform the factor and other models.

As we can see in Figure II, although neither the factor model, the SPF, nor the neural network predict the full depth of the recession, all three show an understanding that the quarter of the Lehman brothers collapse is in a recession regime with GDP forecasts much lower than in the following quarters. Both the AR(2) and the DSGE model show no difference in forecast levels between the

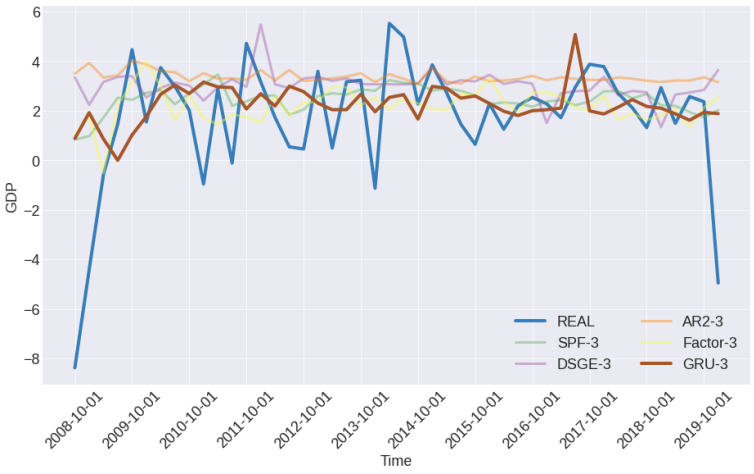

FIGURE II: 3-QUARTERS AHEAD - BASELINE DATA

recession periods and subsequent periods. Additionally, over the expansion period, it seems like the AR(2) and the DSGE are upward biased, which results in even worse performance during the recession[6]. Perhaps it is not surprising that both models cannot forecast recessions as the AR(2) can only look two periods back and the DSGE model has state variables that are relatively unrelated to leading indicators of recessions. For completeness, we also show this same graph tiled for all other horizons (1,2,4,5) below.

Looking at figure III, along the long term horizons, the factor model underperformance seems to be mainly due to a forecast of what seems like a recession bounce back. Here the factor model forecasts a recovery of 9 percent or more which is much larger than the actual recovery.

In addition to these forecast graphs, we provide a Monte Carlo simulation (Table II), estimating our model at each time horizon 100 times. At every horizon, the average root mean squared error of our simulated models indicates competitive, if not outperformance, against baseline models. This is in contrast

6. More information on the biases and variances of the different models can be found in Appendix K.

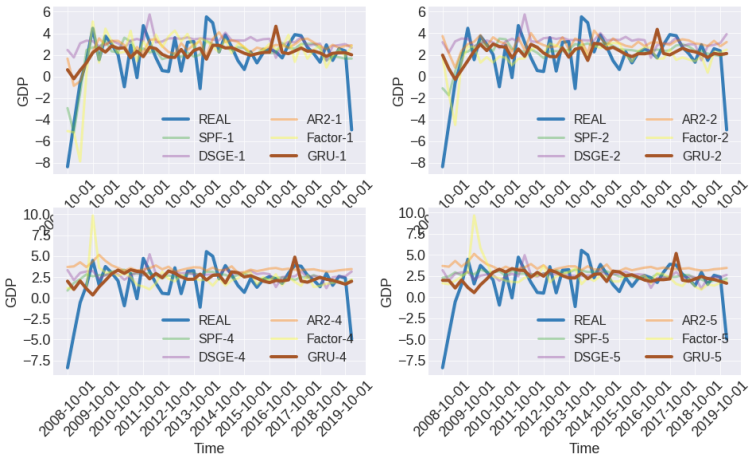

FIGURE III: 1,2,4,5 QUARTERS AHEAD - BASELINE DATA

to the mean forecast in table I which reports the RMSE of the mean forecast rather than the mean of the RMSE forecasts. Interestingly, it seems like the best performing model on validation data, when tested on the test data, often performs worse than the average performance over all the models. This is something that should be investigated further, but based on this phenomenon, we recommend that practitioners take a simple mean or median forecast across many different models instead of picking one model.

TABLE II: BASELINE GRU MONTE CARLO SIMULATION OVER INITIALIZATIONS

THE MEAN AND STANDARD DEVIATION OF THE PERFORMANCE OF GATED RECURRENT UNITS ON THE TEST SET 2009-Q1 TO 2020-Q1

Time (Q's Ahead)	1Q	2Q	3Q	4Q	5Q
Mean RMSE	2.4	2.6	2.5	**2.6**	**2.6***
Std Dev RMSE	0.06	0.06	0.05	0.06	0.06

A common criticism of deep learning attempts at forecasting is that the models are unreliable, but due to high variance, one can p-hack a model that performs well. The Monte Carlo simulation in Table II, anticipates this critique.

The standard deviation of our RMSE is 0.06 which suggests that all our models have a similar performance on the test set when optimization is complete. This cannot resolve all issues, as the Monte Carlo result only deals with numerical instability. The model could still fit this particular data window or architecture choice, due to chance. In order to respond to those critiques, we also provide robustness checks across different architectures and data periods.

One test we preformed was to replace the gated recurrent units with a long short term memory (LSTM) layer (Hochreiter and Schmidhuber, 1997)–another type of recurrent neural network. We use the same test data as the main result (USA 2009-Q1 to 2020-Q1) as well as the same data as inputs. The long short term memory neural network models in Table III are analogous to the gated recurrent unit neural networks models in Table I. Mean RMSE and Std Dev RMSE, correspond to the entries in Table II. The baseline performances are still the same as the test set has not changed.

TABLE III: Baseline LSTM Monte Carlo Simulations

The performance of the best, mean, and median forecasts as well as the mean and standard deviation of the long short term memory networks on the test set 2009-Q1 to 2020-Q1 (lower is better)

Time (Q's Ahead)	1Q	2Q	3Q	4Q	5Q
Best RMSE	2.4	2.6	2.6	**2.6**	**2.6***
RMSE of Mean	2.4	2.6	2.5	**2.6**	**2.6***
RMSE of Median	2.4	2.5	2.5	**2.6**	**2.6***
Mean RMSE	2.4	2.6	2.5	**2.6**	**2.6***
Std Dev RMSE	0.05	0.07	0.05	0.06	0.06

The long short term memory networks outperform the baseline models along essentially the same time horizons. Performance is also competitive, but consistently a little worse, than the gated recurrent unit over all time frames. The LSTM has similar standard deviation of root mean squared error, suggesting that the two models consistently find a similar optimum when it comes to fore-

casting. Again, taking a model average through the mean or median forecast results in small but consistent root mean squared error performance improvements.

Additionally, we re-estimated the model with slightly different test set from 2009-Q4 to 2019-Q4 inclusive as opposed to 2008-Q4 to 2020-Q1, comparing the benchmark economic models to our original gated recurrent unit (Table IV). The reason we use this horizon is that it contains no recessions. Since the highly flexible neural network will have an advantage forecasting periods with a significant departure from a more linear-friendly period of expansion, removing the recessions will hamstring our model compared to the more linear model baselines. Our gated recurrent units were completely re-estimated as we additionally included 2009-Q1 to 2009-Q3 in the validation set. Performance would improve if we left those (recession) time steps out of the validation set as the test set contains no recessions, however, this decision cannot be rationalized from the point-of-view of an out-of-sample forecaster. Although this version of our model does not outperform the best baseline models along any horizon, taking into account performance over all horizons, we think our median and mean models are better than the US AR(2), VAR(1), and the factor model on this test set, while performing slightly worse than the DSGE model and the world AR(2). This supports our hypothesis that the main outperformance of our model was in highly nonlinear domains like recessions and other regime changes. However, using the cross sectional data reduced the tendency for the models to be biased upwards and was a contributor to the gated recurrent neural networks outperformance over models trained only on US data.

This provides supplementary evidence that the outperformance of our neural network is not due to either over-fitting the test set or over-fitting the architecture choice. Additionally, we ran Monte Carlo simulations (Table V) which

TABLE IV: Expansion Root Mean Squared Error

The performance of the best, mean, and median forecasts for the performance of gated recurrent units on the test set 2009-Q4 to 2019-Q4

Time (Q's Ahead)	1Q	2Q	3Q	4Q	5Q
VAR(1)					
US data	2.3	2.6	2.9	3.0	3.0
World data	2.1	2.2	2.2	2.2	2.2
AR(2)					
US data	1.7	1.7	1.8	1.9	1.9
World data	**1.6**	1.6	1.6	**1.5**	**1.5***
Smets Wouters DSGE					
US data	1.8	1.8	1.7	1.6	1.5*
Factor					
US data	1.6	**1.6**	**1.6**	1.9	2.1
GRU*					
Best	1.8	2.3	2.0	2.0	1.9
Mean Forecast	1.7	1.7	1.7	1.7	1.7
Median Forecast	1.7	1.7	1.7	1.7	1.7
SPF Median	1.4	1.5	1.5	1.5	1.5

*All GRU models use entire world data cross section

show that given 100 random initialization and optimization routines over all 5 horizons, the model still consistently achieves low root mean squared error and has a low standard deviation–demonstrating stability and reproducability.

TABLE V: Expansion GRU Monte Carlo Simulations

The mean and standard deviation of the performance of gated recurrent units on the test set 2009-Q4 to 2019-Q4

Time (Q's Ahead)	1Q	2Q	3Q	4Q	5Q
Mean RMSE	1.8	1.8	1.8	1.8	1.7
Std Dev RMSE	0.18	0.18	0.21	0.11	0.08

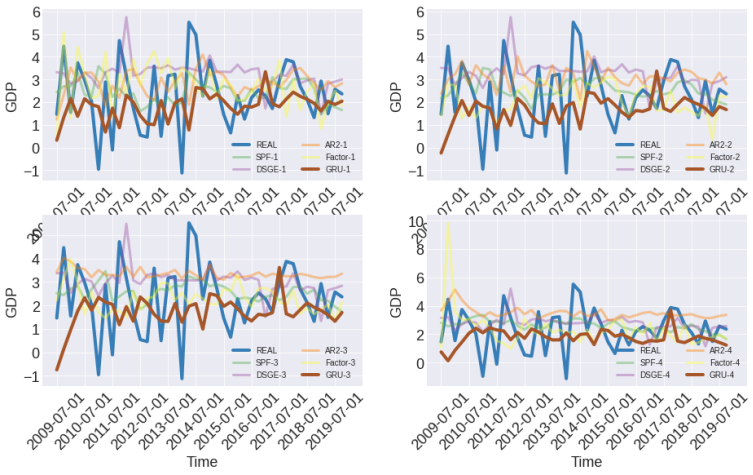

Figure IV: 1,2,3,4 Quarters Ahead - Expansion Data

We also provide regressions analyzing the information content of each forecasting model by regressing true GDP on the forecast models. As none of the models except the SPF have consistent statistically significant information above and beyond the other models, we have placed that analysis in Appendix J. Additionally, we decompose RMSE into bias and variance of each model in Appendix K. In the appendix, we show that the gated recurrent unit seems to have lower bias than all models, except for the factor models. In terms of variance, our model outperforms all models, with the exception of the SPF.

IX. Conclusion

In this paper, we find that a gated recurrent unit recurrent neural network can outperform traditional macroeconomic models for forecasting GDP. We estimate this model by using a cross section of the macroeconomic time series data of 50 countries including the US to help to prevent over-fitting in the neural network. The use of recurrent neural networks and the use of larger macroeconomic data sets to train these models is promising. Clearly, improving the forecasting ability of government institutions and private industry would allow better planning and preparation. For government, this allows monetary and fiscal policy to better anticipate recessions. For industry, this allows businesses to better forecast demand, building or drawing down inventories. Although our model outperforms those commonly used in the literature, there is significant room for further improvement with these models due to the plethora of architecture and data choices for a neural network to use. While we focus on forecasting GDP, the modeling and architecture choices likely extend to other economic variables as well. Additionally, we plan for future work to look into using these models to interrogate economic phenomenon. For example, testing the state space in this model for correlations with economic variables or adding layers like attention which make models easier to interpret (Bahdanau, Cho and Bengio, 2014).

X. Appendix

A. Selected Countries

Countries in data set: Australia, Austria, Belgium, Brazil, Canada, Switzerland, Chile, Columbia, Cyprus, Czech Republic, Germany, Denmark, Spain, Estonia, European Union, Finland, France, Great Britain, Greece, Hong Kong, Croatia, Hungry, Ireland, Israel, Italy, Japan, Korea, Luxembourg, Latvia, Mexico, Mauritius, Malaysia, Netherlands, Norway, New Zealand, Peru, Philippines, Poland, Portugal, Romania, Russia, Singapore, Slovakia, Slovenia, Sweden, Thailand, Turkey, USA and South Africa.

B. Selected Performance: Graphs

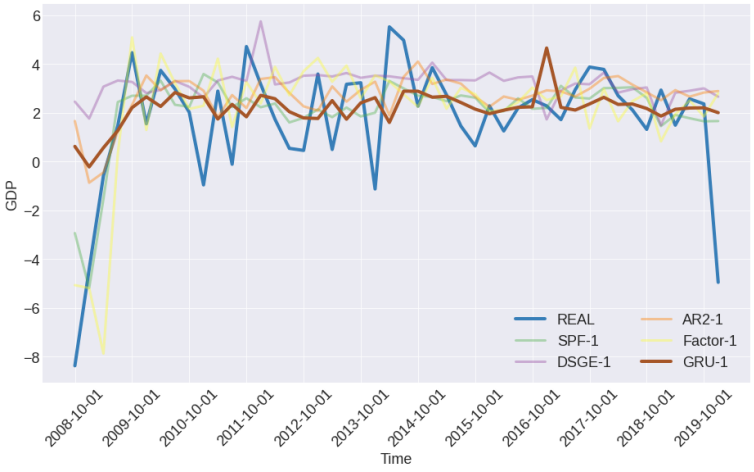

Figure V: 1-Quarter Ahead - Baseline Data

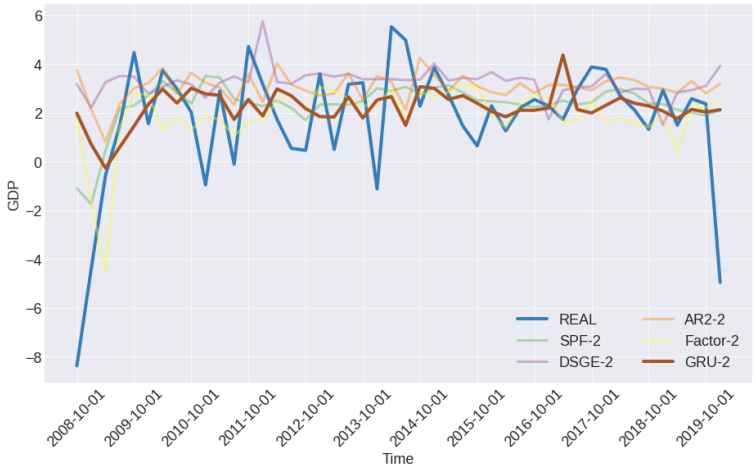

Figure VI: 2-Quarters Ahead - Baseline Data

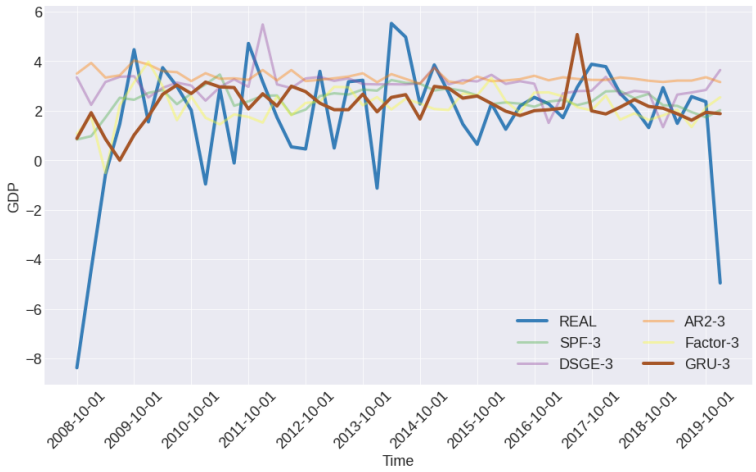

Figure VII: 3-Quarters Ahead - Baseline Data

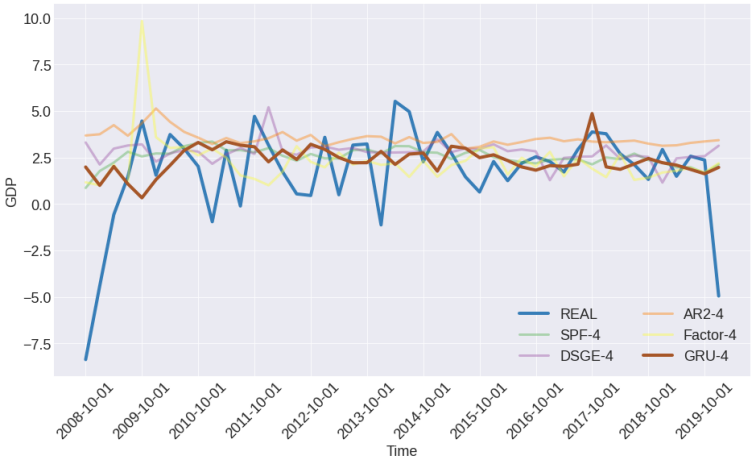

FIGURE VIII: 4-QUARTERS AHEAD - BASELINE DATA

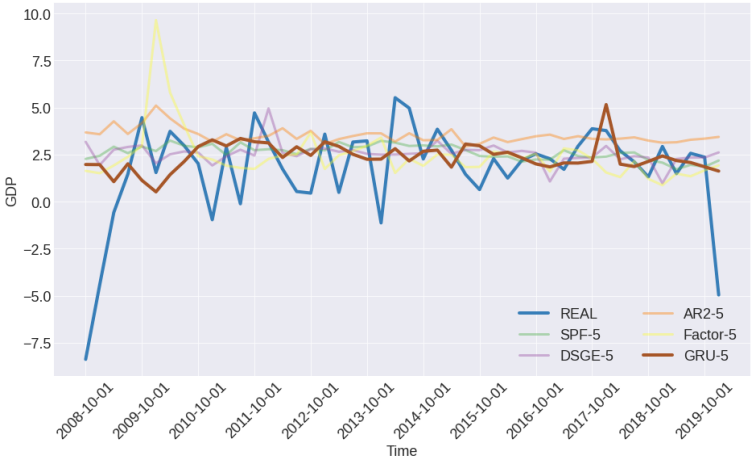

FIGURE IX: 5-QUARTERS AHEAD - BASELINE DATA

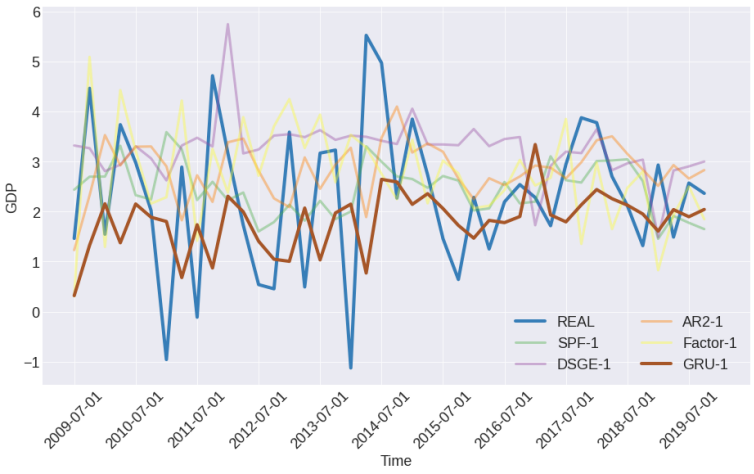

Figure X: 1-Quarter Ahead - Expansions Data

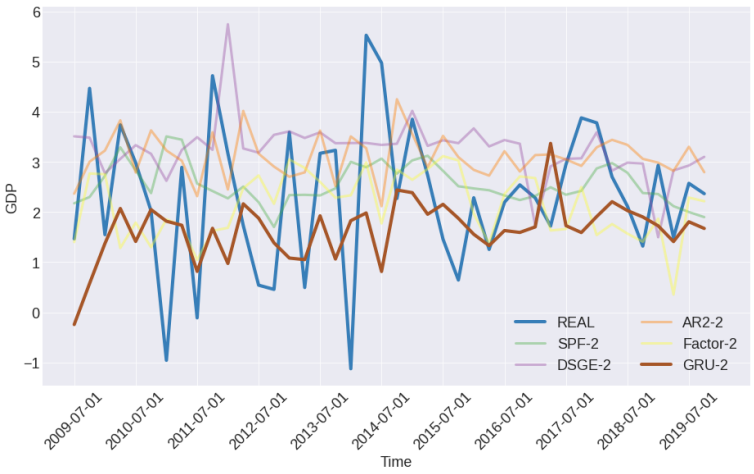

Figure XI: 2-Quarters Ahead - Expansions Data

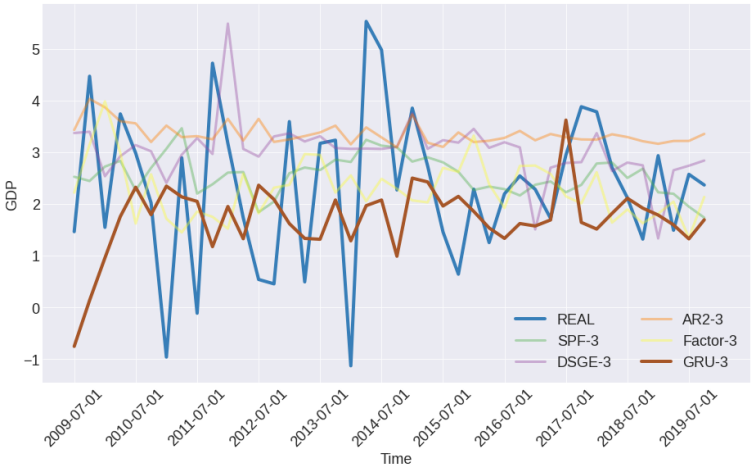

Figure XII: 3-Quarters Ahead - Expansions Data

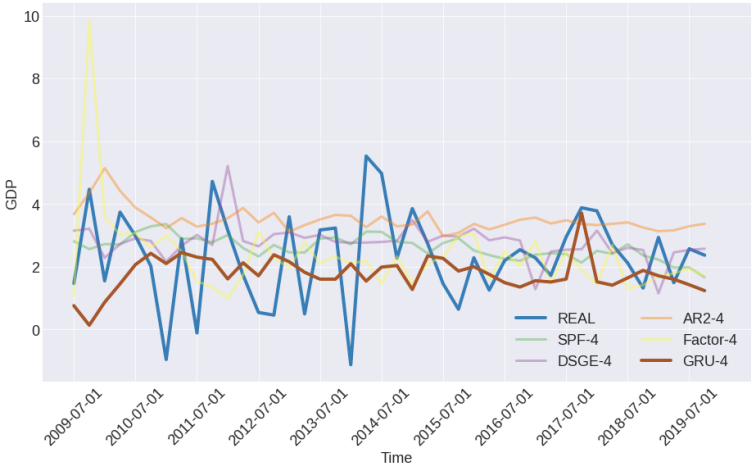

Figure XIII: 4-Quarters Ahead - Expansions Data

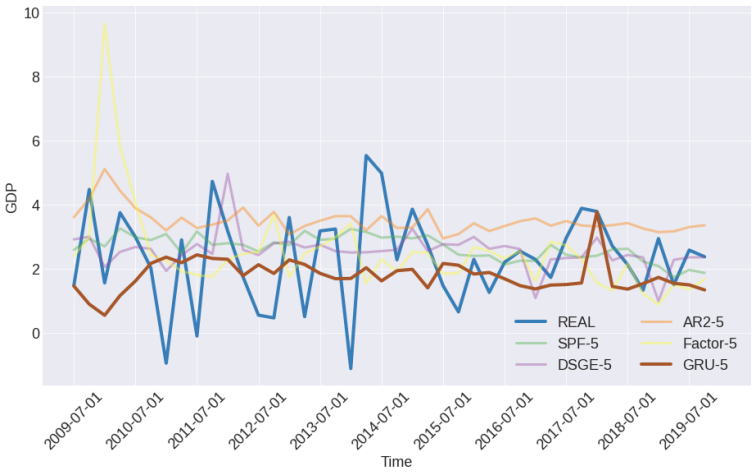

FIGURE XIV: 5-QUARTERS AHEAD - EXPANSIONS DATA

C. Estimating the Parameters of a Linear State Space Model on Data

Going back to the two equations that define the linear state space model:

$$Y_t = D * S_t + E * Y_{t-1} + F, \tag{14}$$

$$S_t = A * S_{t-1} + B * Y_{t-1} + C \tag{15}$$

We use equations 12 and 13 to recursively substitute for the model prediction at a particular time period. The forecast for period 1 then is:

$$\hat{y}_1 = D * (A * 0 + B * Y_0 + C) + E * Y_0 + F \tag{16}$$

and the forecast for period 2 is:

$$\hat{y}_2 = D * (A * (A * 0 + B * Y_0 + C) + B * Y_1 + C) + E * Y_1 + F \tag{17}$$

Hatted variables indicate predictions and un-hatted variables correspond to ac-

tual data. Additional time periods would be solved by iteratively substituting for the state using equation 12 and 13 for the previous state. In order to update the parameters matrices A, B, C, D, E, and F, the gradient is derived for each matrix and each parameter is updated via hill climbing. We will illustrate the process of hill climbing by taking the gradient of one parameter B:

$$\frac{\partial \sum_{\forall t} L(y - \hat{y})}{\partial B} = \frac{\partial L(y_1 - \hat{y}_1)}{\partial B} + \frac{\partial L(y_2 - \hat{y}_2)}{\partial B} \tag{18}$$

Here $L()$ indicates the loss function. Substituting for y_1' and y_2' with equations 14 and 15 into 16 and using squared error as the loss function, we arrive at an equation with which we can take partial derivatives for with respect to A:

$$\frac{\partial}{\partial B} L = \frac{\partial}{\partial B} \frac{1}{2} (y_1 - D * (A * 0 + B * Y_0 + C) + E * Y_0 + F)^2 \tag{19}$$

$$+ \frac{\partial}{\partial B} \frac{1}{2} (y_2 - D * (A * (A * 0 + B * Y_0 + C) + B * Y_1 + C) + E * Y_1 + F)^2 \tag{20}$$

Distributing all the B's and taking the derivative of X. results in $\frac{\partial}{\partial B} L = -(y_1 - D * (A * 0 + B * Y_0 + C) + E * Y_0 + F) * D * Y_0 - (y_2 - D * (A * (A * 0 + B * Y_0 + C) + B * Y_1 + C) + E * Y_1 + F) * (D * A * Y_0 + D * Y_1)$ which provide the gradients for hill climbing. In practice, the derivatives are taken automatically in code.

D. The Gated Recurrent Unit in Depth

In the gated recurrent unit, the first equation is typically called the update gate:

$$\mathbf{z}_t = \sigma(\mathbf{x}_t \mathbf{U}^z + \mathbf{h}_{t-1} \mathbf{W}^z) \tag{21}$$

The update gate modifies how much new information (in the form of $\mathbf{x}_t$ and $\mathbf{h}_t$) changes the state vector values. $\mathbf{h}_{t-1}$ is the output of the gated recurrent unit equation for the previous time step, which will be discussed below (see equation 22 for the definition of h). This model allows the internal state to gradually "forget" irrelevant information as well as integrate in new information. In this way, the gate can prevent the state vector $\mathbf{h}$ from exploding, as is a problem with the linear state space model. The next equation defines the reset gate:

$$\mathbf{r}_t = \sigma(\mathbf{x}_t\mathbf{U}^r + \mathbf{h}_{t-1}\mathbf{W}^r) \tag{22}$$

The reset gate is used as an additional forget gate for the input data and the state at the previous time step is aggregated into the candidate activation vector:

$$\mathbf{s}_t = \tanh(\mathbf{x}_t\mathbf{U}^s + (\mathbf{h}_{t-1} \odot \mathbf{r}_t)\mathbf{W}^s) \tag{23}$$

The final equation is output vector which is the state input of the next time step:

$$\mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{s}_t + \mathbf{z}_t \odot \mathbf{h}_{t-1} \tag{24}$$

This equation takes a weighted average with the weight determined by the update gate of the previous state $\mathbf{h}_{t-1}$ and the newly created candidate $\mathbf{s}_t$ which consists of the new information (e.g. $\mathbf{x}_t$) as well as the previous state. Typically one more equation of a linear or nonlinear form would convert the state variable to the output.

E. The Rectified Linear Unit

A non-linearity used in our architecture, but not in the gated recurrent unit layers is the rectified linear unit (ReLU) (Agarap, 2018). The rectified linear unit is defined as:

$$ReLU(x) = max(0, x) \tag{25}$$

The ReLU is the identity operation with a floor of zero much like the payoff of a call option. Despite being almost the identity map, this non-linearity applied in a large enough neural network can approximate any function (Hornik, Stinchcombe and White, 1989).

F. Skip Connections and Batch Norm

Skip connections (He et al., 2015) allow the input to skip the operation in a given layer. The input is then just added onto the output of the skipped layer, forming the final output of the layer. This allows the layer being skipped to learn a difference between the "correct" output and input, instead of learning a transformation. Additionally, if the model is overfitting, the neural network can learn the identity map easily. Skip connections are used when the input and the output are the same dimension which allows each input to correspond to one output. Because our network does not have this property, we learn a linear matrix that converts to the input to the output dimension. All the skip connections are linear operations and have no activation or batch norm, which differs from the pair of dense layers at the beginning of the network which have both batch norm and rectified linear unit activations.

Batch normalizing (Ioffe and Szegedy, 2015) is used to prevent drift of output through a deep neural network. Changes to parameters in the early layers will

cause an out-sized effect on the output values for the later layers. Batch norm fixes this problem by normalizing the output to look like a standard normal distribution after the output of each layer. Thus the effect of changes in parameters will not greatly effect the magnitude of output vector as between each output the data is re-normalized to have a mean of 0 and a standard deviation on 1.

G. Details on the Survey of Professional Forecasters

While our model uses the 2020 vintage of data, in reality, the forecasters for the Survey of Professional Forecasters were working with pseudo-out-of-sample vintages when forecasting over the entire test. While reproducing this would be possible from using old vintages, this would require estimating the model at every time step of the test set as the data would change every period. We wanted to avoid this pseudo-out-of-sample forecasting as it would result in estimating 4600 models instead of 100 at each horizon. Furthermore, the benefit of this increased computation is not clear as we would still be using 2020 vintage data for countries outside the US, many of which old vintages are difficult or impossible to find. As such, it was easier to compare the SPF performance on the 2020 vintage. Furthermore, all our baselines models are estimated and evaluated using the 2020 vintage as well and so this choice allows one to compare the SPF performance with the performance of all the baseline models.

H. Detailed Description of Economic Baseline Models

A first model we use is the autoregressive model, AR(n). An often used benchmark model, it estimates a linear relationship using the independent variable lagged n times. In terms of forecasting ability, this model is competitive with or outperforms the other economic models in our tests which is consis-

tent with (Diebold, 1998). We use a autoregressive model with two lags and a constant term.

Additionally, we compared the Smets-Wouters 2007 model (Smets and Wouters, 2007), as DSGE models share many similarities with recurrent neural networks and the Smets-Wouters paper cited directly above suggests that this particular model can outperform VARs and BVARs in forecasting. When running this, we use the standard Smets-Wouters Dynare code contained on the the published paper's data appendix. For computational reasons, we estimate the model using Laplace's approximation to the optimal Bayesian posterior and then forecast based on the posterior of the deep parameters.

A final model we included in our baseline economic models were factor models (see Stock and Watson (2002a) and Stock and Watson (2002b)). While these models were the most effective especially at lower horizons, these models are also dependent on a large cross section of economic data with a long history in a country. In reality, only a few other developed countries have a cross section of data that would be large enough to permit using these models as effectively as can be used in the United States. That being said, factor models do outperform our neural networks at shorter time intervals and we imagine there is promise combining the factor approach with a neural network approach. In short, the factor model approach takes a large cross section of data and uses a technique like principal components analysis to reduce the dimensionality of the problem. In our case, we concatenate 5-8 principle components based on information criterion of the high dimensional data with a lagged value of GDP and regress future GDP on this. We modify and use the code from FRED-QD as our baseline factor model (McCracken and Ng, 2016).

While we considered the forecasting performance of vector autoregressions (Sims, 1980), these models did not perform well enough to report in our main

tables. However, because of available cross country data, we compared this model and the AR(2) with our gated recurrent unit on our 50 countries cross section test. Since we are only forecasting GDP, the vector autoregressive models use lagged GDP, consumption and unemployment used to forecast the single GDP variable as a linear regression.

All the economic models were estimated on US GDP as is standard. While we ran preliminary test on estimating these models on our cross section of 50 countries, we ran into issues with estimating both factor models and DSGE models this way. However, preliminary results on the AR(2) model suggests there could be some improvement to using a cross section even on a 3 parameter AR(2) model. The improvement is not as large as the GRU which is not surprising as the GRU has more parameters to take advantage of a larger data set.

I. Adam Optimizer

Adam combines momentum (Polyak, 1964)–a technique that uses recent history to smooth out swings orthonormal to the objective direction–with RMSprop (Tieleman and Hinton, 2012)–a technique used to adjust step size based on gradient volatility.

Traditional gradient descent hill climbing updates the parameters with a single equation:

$$\theta_t = \theta_{t-1} - \lambda * \nabla_\theta L_\theta(x, y) \tag{26}$$

Here $\nabla_\theta L_\theta(x, y)$ denotes taking the gradient of the loss with respect to θ, the parameters of the model. For convenience, I will denote this term g_t. By subtracting the gradient multiplied by a small step size λ, this is moving the parameters theta in the direction the reduces the loss the most at θ_{t-1}

If we wanted to use information from the second derivative to inform optimization, we can use Newton-Raphson instead:

$$\theta_t = \theta_{t-1} - H_t^{-1} * g_t \tag{27}$$

This uses the Hessian to determine an optimal step size based on steepness in the loss function. Typically, this approach is not used in deep learning as deep learning models typically have a large number of parameters and calculating the Hessian has a quadratic cost in the number of parameters and inverting also has a super-linear cost. However, there are quasi-newton methods that attempt to approximate the Hessian to determine the step size without the high computational cost. Adam is similar to these methods. The equations that define Adam are as follows:

$$\nu_t = \beta_1 * \nu_{t-1} - (1 - \beta_1) g_t \tag{28}$$

$$s_t = \beta_2 * s_{t-1} - (1 - \beta_2) * g_t^2 \tag{29}$$

$$\delta\theta_t = -\eta \frac{\nu_t}{\sqrt{s_t + \epsilon}} * g_t \tag{30}$$

$$\theta_{t+1} = \theta_t + \delta\theta_t \tag{31}$$

The first equation is a moving average of the gradient. This "momentum" term is used because often in training the direction of the gradient would move nearly perpendicular to the direction towards the optimum. Gradient descent would spend a lot of time zig-zaging while only making slow progress towards an optimum (see figure XV). Taking a moving average of previous gradients preserves the principle direction while the orthogonal directions cancel each other out.

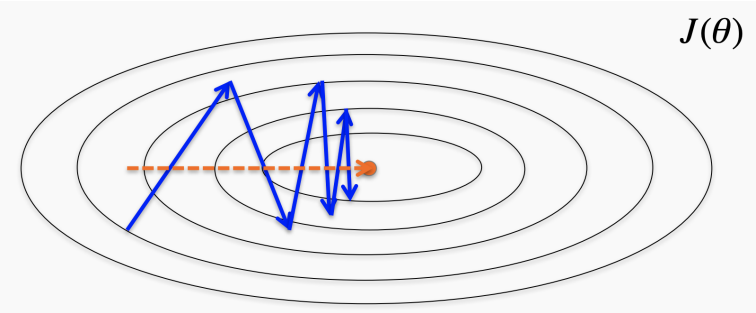

FIGURE XV: MOMENTUM

Likewise the s_t equation which is a moving average approximation for the Hessian. The approximate Hessian is used for adjusting the step size of the algorithm based on the curvature of the loss function at a given point. β_1 and β_2 are hyperparameters that determine the smoothness of the moving average. Again the resulting update term is applied to the previous values of the parameters. This approach is empirically shown to lead to more stable optimization and even better optimia than simpler gradient descent approaches for large networks.

J. Information Content Regressions

We regress true GDP on a varying collection of forecasts to test for statistically significant contribution of a given forecast like our gated recurrent unit model. An interpretation of significant coefficients would be that the given forecast method is adding statistically significant information when pooled with the other regressions.

Here is the median gated recurrent unit forecast compared to the SPF on the baseline test set ranging from 1-quarter ahead forecasts to 5-quarters ahead:

TABLE VI

	Real GDP Growth				
	(1-Qtr)	(2-Qtrs)	(3-Qtrs)	(4-Qtrs)	(5-Qtrs)
GRU	−0.195	−0.653	0.277	−0.072	0.505
	(0.541)	(0.462)	(0.414)	(0.471)	(0.517)
SPF	1.210***	2.064***	3.050***	2.957***	1.023
	(0.249)	(0.350)	(0.619)	(0.765)	(1.002)
Constant	−0.339	−1.635*	−6.370***	−5.592***	−2.161
	(0.952)	(0.953)	(1.557)	(1.970)	(2.824)
N	46	46	46	46	46
R^2	0.484	0.474	0.404	0.270	0.049
Adjusted R^2	0.460	0.449	0.377	0.236	0.005
Resid. Std. Error (df = 43)	1.880	1.898	2.020	2.237	2.553
F Statistic (df = 2; 43)	20.157***	19.367***	14.587***	7.944***	1.103

Notes: ***Significant at the 1 percent level.
**Significant at the 5 percent level.
*Significant at the 10 percent level.

The following table contains regressions comparing the information content of all the forecasting models (the GRU and baselines) and excluding the SPF:

TABLE VII

	Real GDP Growth				
	(1-Qtr)	(2-Qtrs)	(3-Qtrs)	(4-Qtrs)	(5-Qtrs)
GRU	1.489*	1.300	0.810	0.613	0.794
	(0.757)	(0.837)	(0.495)	(0.562)	(0.732)
DSGE	0.918*	0.137	0.168	0.055	−0.209
	(0.514)	(0.631	(0.649)	(0.675)	(0.741)
AR2	−1.139*	−1.998*	−0.645	−0.619	−0.466
	(0.640)	(1.125)	(1.768)	(1.108)	(1.922)
Factor	0.572***	0.710*	0.837	0.548*	0.440
	(0.176)	(0.388)	(0.542)	(0.325)	(0.461)
Constant	−2.655	3.156	−0.250	1.075	0.945
	(1.913)	(3.290)	(6.192)	(4.656)	(6.938)
N	46	46	46	46	46
R^2	0.444	0.158	0.120	0.080	0.057
Adjusted R^2	0.390	0.076	0.034	−0.010	−0.035
Resid. Std. Error (df = 41)	1.998	2.460	2.514	2.571	2.603
F Statistic (df = 4; 41)	8.198***	1.924	1.400	0.889	0.619

Notes:

***Significant at the 1 percent level.
**Significant at the 5 percent level.
*Significant at the 10 percent level.

This final table performs the same regression but has the SPF, GRU, and all baseline models:

TABLE VIII

	Real GDP Growth				
	(1-Qtr)	(2-Qtrs)	(3-Qtrs)	(4-Qtrs)	(5-Qtrs)
GRU	0.979	−0.332	0.277	−0.050	0.607
	(0.697)	(0.743)	(0.426)	(0.530)	(0.776)
SPF	1.140***	2.004***	3.112***	2.907***	0.903
	(0.345)	(0.404)	(0.698)	(0.818)	(1.189)
DSGE	0.661	−0.101	−0.211	−0.336	−0.301
	(0.468)	(0.504)	(0.544)	(0.606)	(0.755)
AR2	−1.408**	−0.575	0.915	−0.816	−0.829
	(0.580)	(0.941)	(1.505)	(0.979)	(1.990)
Factor	0.153	0.075	0.073	0.306	0.394
	(0.203)	(0.334)	(0.480)	(0.295)	(0.467)
Constant	−1.443	−0.270	−9.119	−2.398	0.635
	(1.755)	(2.710)	(5.498)	(4.225)	(6.986)
N	46	46	46	46	46
R^2	0.564	0.479	0.412	0.301	0.070
Adjusted R^2	0.509	0.414	0.339	0.213	−0.046
Residual Std. Error (df = 40)	1.793	1.959	2.081	2.270	2.617
F Statistic (df = 5; 40)	10.331***	7.356***	5.606***	3.438**	0.606

Notes:

***Significant at the 1 percent level.
**Significant at the 5 percent level.
*Significant at the 10 percent level.

K. Bias and Variance in the Forecasting Models

The following table contains the mean bias as well as the standard error and an indication of significance of that bias. The standard errors of the residual double as the (square-root) of the variance estimate. For the gated recurrent unit we use the median forecast:

TABLE IX

	Forecast Bias				
	(1-Qtr)	(2-Qtrs)	(3-Qtrs)	(4-Qtrs)	(5-Qtrs)
GRU	0.459*	0.480*	0.506*	0.620*	0.644**
	(0.343)	(0.369)	(0.365)	(0.379)	(0.375)
SPF	0.331	0.600**	0.723**	0.804**	0.901***
	(0.274)	(0.302)	(0.335)	(0.347)	(0.372)
DSGE	1.459***	1.513***	1.300***	1.058***	0.827***
	(0.354)	(0.378)	(0.544)	(0.386)	(0.385)
AR2	0.937***	1.317***	1.636***	1.795***	1.780***
	(0.351)	(0.381)	(0.380)	(0.383)	(0.384)
Factor	0.432*	0.163	0.459	0.533*	0.699**
	(0.328)	(0.449)	(0.367)	(0.390)	(0.414)

Notes:
***Significant at the 1 percent level.
**Significant at the 5 percent level.
*Significant at the 10 percent level.

Significance indicates that the forecasting model is biased in a statistical sense. Including the SPF, the above table suggests that the gated recurrent unit derives much of its outperformance due to less biased forecasts, only slightly more biased than a factor model. The factor model, though, has significant variance as illustrated on the graphs in Appendix B or gleaned from the standard deviations in the above table.

L. Diebold-Mariano Test for Statistically Significant Forecasting Outperformance

We compared our forecasts using the Diebold-Mariano test (Diebold and Mariano, 2002), at each horizon, to the best performing economic baseline. The Diebold-Mariano tests for statistically significant forecasting outperformance using a comparison of forecast residuals. These tests were not significant at the 10 percent level for each horizon independently, although the two horizon test outperformed with a p-value under 20 percent and the model outperformed certain non-optimal baseline models in a statistically significant way. As these results have limited significance across the board, we have chosen not to display this information in our charts.

References

Agarap, Abien Fred. 2018. "Deep Learning using Rectified Linear Units (ReLU)." *CoRR*, abs/1803.08375.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. 2014. "Neural machine translation by jointly learning to align and translate." *arXiv preprint arXiv:1409.0473*.

Baltagi, Badi H. 2008. "Forecasting with panel data." *Journal of forecasting*, 27(2): 153–173.

Cho, KyungHyun, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. "On the Properties of Neural Machine Translation: Encoder-Decoder Approaches." *CoRR*, abs/1409.1259.

Chung, Junyoung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. "Empirical evaluation of gated recurrent neural networks on sequence modeling." *arXiv preprint arXiv:1412.3555*.

Crandall, Robert W, William Lehr, and Robert E Litan. 2007. "The effects of broadband deployment on output and employment: A cross-sectional analysis of US data."

Diebold, Francis X. 1998. *Elements of forecasting*. South-Western College Pub.

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

Edge, Rochelle M, Michael T Kiley, and Jean-Philippe Laforte. 2010. "A comparison of forecast performance between federal reserve staff forecasts, simple reduced-form models, and a DSGE model." *Journal of Applied Econometrics*, 25(4): 720–754.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. "Deep Residual Learning for Image Recognition."

Hochreiter, Sepp, and Jürgen Schmidhuber. 1997. "Long Short-Term Memory." *Neural Comput.*, 9(8): 1735–1780.

Hochreiter, Sepp, Martin Heusel, and Klaus Obermayer. 2007. "Fast model-based protein homology detection without alignment." *Bioinformatics*, 23(14): 1728–1736.

Hornik, Kurt, Maxwell Stinchcombe, and Halbert White. 1989. "Multilayer feedforward networks are universal approximators." *Neural Networks*, 2(5): 359 – 366.

Ioffe, Sergey, and Christian Szegedy. 2015. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift."

Kaplan, Jared, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. "Scaling laws for neural language models." *arXiv preprint arXiv:2001.08361*.

Kingma, Diederik P., and Jimmy Ba. 2014. "Adam: A Method for Stochastic Optimization."

Koop, Gary, Roberto Leon-Gonzalez, and Rodney Strachan. 2012. "Bayesian model averaging in the instrumental variable regression model." *Journal of Econometrics*, 171(2): 237–250.

Leamer, Edward E, and Edward E Leamer. 1978. *Specification searches: Ad hoc inference with nonexperimental data.* Vol. 53, Wiley New York.

Li, Xiaohua, Weijin Zhuang, and Hong Zhang. 2020. "Short-term Power Load Forecasting Based on Gate Recurrent Unit Network and Cloud Computing Platform." 1–6.

Mayer, H., F. Gomez, D. Wierstra, I. Nagy, A. Knoll, and J. Schmidhuber. 2006. "A System for Robotic Heart Surgery that Learns to Tie Knots Using Recurrent Neural Networks." 543–548.

McCracken, Michael W, and Serena Ng. 2016. "FRED-MD: A monthly database for macroeconomic research." *Journal of Business & Economic Statistics*, 34(4): 574–589.

Minh, Dang Lien, Abolghasem Sadeghi-Niaraki, Huynh Duc Huy, Kyungbok Min, and Hyeonjoon Moon. 2018. "Deep learning approach for short-term stock trends prediction based on two-stream gated recurrent unit network." *Ieee Access*, 6: 55392–55404.

Miyamoto, Wataru, Thuy Lan Nguyen, and Dmitriy Sergeyev. 2018. "Government spending multipliers under the zero lower bound: Evidence from Japan." *American Economic Journal: Macroeconomics*, 10(3): 247–77.

None. 1968. "Survey of Professional Forecasters." *https://www.philadelphiafed.org/research-and-data/real-time-center/survey-of-professional-forecasters*, Accessed: 2020-09-01.

Panzner, Maximilian, and Philipp Cimiano. 2016. "Comparing hidden markov models and long short term memory neural networks for learning action representations." 94–105, Springer.

Pesaran, M Hashem, and Allan Timmermann. 1992. "A simple nonparametric test of predictive performance." *Journal of Business & Economic Statistics*, 10(4): 461–465.

Polyak, Boris. 1964. "Some methods of speeding up the convergence of iteration methods." *Ussr Computational Mathematics and Mathematical Physics*, 4: 1–17.

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams. 1986. "Learning representations by back-propagating errors." *nature*, 323(6088): 533–536.

Shi, Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun Woo. 2015. "Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting." In *Advances in Neural Information Processing Systems 28.* , ed. C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama and R. Garnett, 802–810. Curran Associates, Inc.

Sims, Christopher A. 1980. "Macroeconomics and Reality." *Econometrica*, 48(1): 1–48.

Smets, Frank, and Rafael Wouters. 2007. "Shocks and frictions in US business cycles: A Bayesian DSGE approach." *The American Economic Review*, 97(3): 586–606.

Stock, James H, and Mark W Watson. 2002*a*. "Forecasting using principal components from a large number of predictors." *Journal of the American statistical association*, 97(460): 1167–1179.

Stock, James H, and Mark W Watson. 2002*b*. "Macroeconomic forecasting using diffusion indexes." *Journal of Business & Economic Statistics*, 20(2): 147–162.

Tieleman, T., and G. Hinton. 2012. "Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude." *COURSERA: Neural Networks for Machine Learning*.

Walker, Gilbert Thomas. 1931. "On periodicity in series of related terms." 131(818): 518–532.

Watson, MW. 2001. "Time series: economic forecasting." *International Encyclopedia of the Social & Behavioral Sciences*, 15721–15724.

West, Kenneth D. 1996. "Asymptotic inference about predictive ability." *Econometrica: Journal of the Econometric Society*, 1067–1084.

Zerroug, A, L Terrissa, and A Faure. 2013. "Chaotic dynamical behavior of recurrent neural network." *Annu. Rev. Chaos Theory Bifurc. Dyn. Syst*, 4: 55–66.

Department of Economics, University of Michigan, Ann Arbor