Forecasting GDP with Recurrent Neural Networks

Cameron Fen and Samir Undavia

December 3rd, 2020

Background and Contribution

- Forecasting is important for policymakers and private industry
- Good predictive models can lead to a deeper understanding of economics primitives
- Our neural network outperforms state-of-the-art economic forecasting models for GDP
  - Also outperforms the Median Survey of Professional Forecasters at 5 Quarters ahead*

What We do (Continued)

- We estimate this model using a cross section of countries in addition to US data
- A cross sectional dimension gives us power to estimate our recurrent neural network model with 17,000 parameters
- We also demonstrate that a single model trained on this cross section is able to generalize across policy regimes/countries

The Data and Baseline Models

- We use growth rate data from 50 different developed countries (GDP, consumption, unemployment) to forecast GDP growth
- Data Sources: World Bank and Trading Economics via Quandl
- Baseline Comparison Models
  1. AR(2)
  2. Smets Wouters 2007 DSGE
  3. Factor Model
  4. SPF*

Baseline Models: AR(2)

- We use a linear model which has two lags of GDP to forecast ahead
- Despite the simplicity, this is one of the workhorse models among forecasting practitioners (Hamilton 1994)
- We find that the AR(2) outperforms the Smets Wouters DSGE at shorter time intervals (1-2 quarters ahead), but is outperformed by Smets Wouters at longer intervals (4-5 quarters ahead)
- A factor model augmented with one GDP lag seems to dominate the AR(2)

Baseline Models: DSGE (Smets and Wouters 2007)

- Smets Wouters is New Keynesian DSGE model that is essentially an extension of the Christiano et. al. 2005 model estimated in a Bayesian framework
- The model is geared towards forecasting rather than macroeconomic analysis
- Despite limited attention paid by practitioners, we think this model deserves more attention, especially at longer time horizons
- The model outperforms AR(2) and factor models at longer horizons, but is unable to detect the great recession at shorter horizons which leads to under-performance

Baseline Models: Factor Models

- These models were introduced by Stock and Watson 2002 and Forni et. al. 2000
- These models take a large cross section of data (in our case 248 data series) in order to forecast and PCA regression, in our case, reduce the large cross section to 8 factors
- We extend the factor model highlighted in Fred-QD (McCraken and Ng 2020) by combining factors estimated in a pseudo-out-of-sample manner along with a lag of GDP for forecasting
- This model is the most formidable competitor to the neural network, however, it under-performs over long horizons likely because of variance issues and the limited predictive power of the cross section of variables

Our Model

- We use a Recurrent Neural Network (RNN) as our forecast model
- An RNN is a state space model of which the simplest is a linear state space model like a Kalman filter
- It contains: A state equation,

$$S_t = AS_{t-1} + BX_{t-1} + C + \epsilon_t \quad (1)$$

- ...and a measurement equation

$$X_t = DS_t + EX_{t-1} + F + \epsilon_t \quad (2)$$

Exploding and Vanishing States

- if we write s_{t+n} as a function of s_t, we see that

$$s_{t+n}(s_t) = A^n s_t + o(s_t) \tag{3}$$

- If n is a large number, depending on the eigenvalues of A, s_{t+n} will either become unbounded for large n or returns to a steady state that doesn't involve s_t
- With RNNs you want to build a model that can remember long term dependencies between data, but also won't have a state that explodes or returns to a steady state

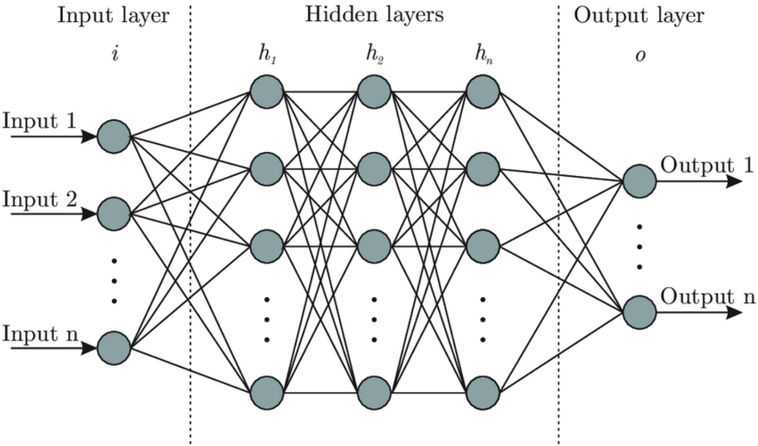
Gates

- The solution to this: Gates
- A gate is a logistic regression with $\sigma(x)$ denoting the logistic link function: $\frac{1}{1+e^{-x}}$
- These gates output a number between 0-1 and are conditioned on the input data, the previous prediction, and the state
- The gates tell the model how much of the old state to forget and how much of the new information to remember
- The gate can dynamically adjust to keep exploding gradients bounded
- With gates, the asymptotic behavior of states can be periodic or even chaotic (Zerroug, 2013), allowing information to be held in the states for a longer period of time

A GRU Cell

$$
\begin{aligned}
z_t &= \sigma_g(W_z x_t + U_z h_{t-1} + b_z) \\
r_t &= \sigma_g(W_r x_t + U_r h_{t-1} + b_r) \\
\hat{h}_t &= \phi_h(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) \\
h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t
\end{aligned}
$$

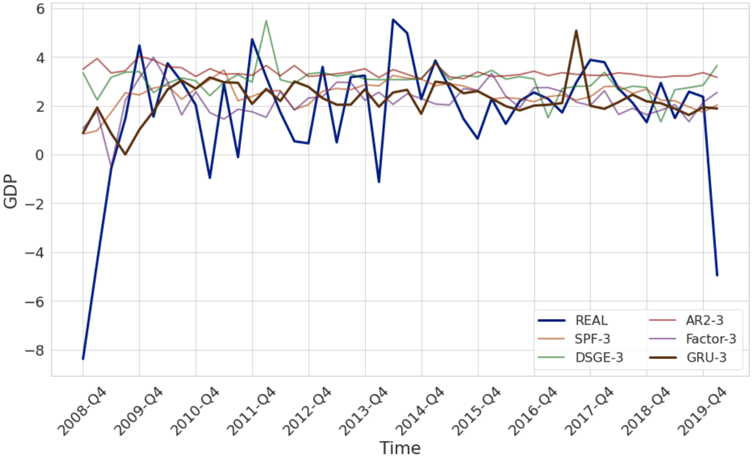
Dense Layers: A Picture



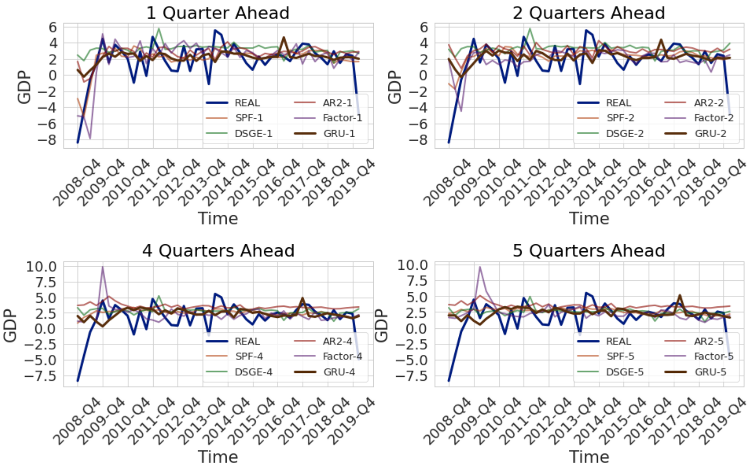
Architecture Structure

- Dense nets and dense layers are the prototypical neural network
- Although the GRU cell is the centerpiece of our model, we also use dense layers to “pre-proces” the data before feeding into the GRU
    - “Pre-proces” is a analogy–the whole network is trained end-to-end
- Additionally we use skip connections (He et. al. 2015) and Batch Normalization (Ioffe et. al. 2015) between layers as well as the Adam optimizer (Kingma 2014)

Graph of Forecast Performance: 3 Quarters Ahead



Graph of Forecast Performance



Out of Sample Forecast Comparison: GRU Model

Table 1: GRU RMSE Performance

Time (Q's Ahead)	1Q	2Q	3Q	4Q	5Q
Smets Wouters DSGE	2.8	3.0	2.9	2.8	2.7
AR-2	2.5	2.9	3.0	3.1	3.1
Factor	**2.2**	**2.5**	2.5	2.7	2.9
GRU Model (Best)	2.4	2.5	2.6	**2.6**	**2.6***
GRU Model (Mean Forecast)	2.3	2.5	**2.5**	**2.6**	**2.6***
GRU Model (Median Forecast)	2.3	2.5	**2.5**	**2.6**	**2.6***
SPF Median	1.9	2.1	2.4	2.5	2.7

Out of Sample Forecast Comparison: GRU Monte Carlo

Table 2: GRU Monte Carlo Simulations

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

Out of Sample Forecast Comparison: LSTM Model

Table 3: LSTM Forecast Performance

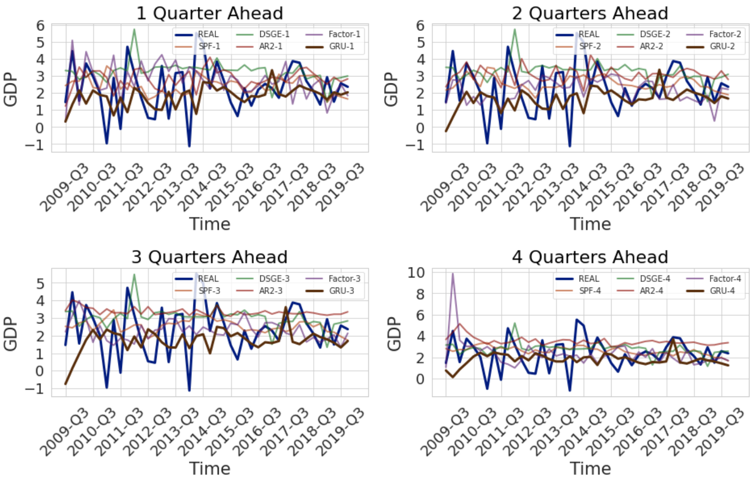
Time (Q's Ahead)	1Q	2Q	3Q	4Q	5Q
Best RMSE	2.4	2.6	2.6	**2.6**	**2.6***
RMSE of Mean	2.4	2.6	2.5	**2.6**	**2.6***
RMSE of Median	2.4	2.5	2.5	**2.6**	**2.6***
Mean RMSE	2.4	2.6	2.5	**2.6**	**2.6***
Std Dev RMSE	0.05	0.07	0.05	0.06	0.06

Out of Sample Forecast Comparison: Forecast of Expansions

Table 4: RMSE Performance on Expansions

Time (Q's Ahead)	1Q	2Q	3Q	4Q	5Q
Smets Wouters DSGE	1.8	1.8	1.7	**1.6**	**1.5***
AR(2)	1.7	1.7	1.8	1.9	1.9
Factor	**1.6**	**1.6**	**1.6**	1.9	2.1
GRU Model (Best)	1.8	2.3	2.0	2.0	1.9
GRU Model (Mean Forecast)	1.7	1.7	1.7	1.7	1.7
GRU Model (Median Forecast)	1.7	1.7	1.7	1.7	1.7
SPF Median	1.4	1.5	1.5	1.5	1.5

Graph of Forecast Performance



Worldwide GRU Model Performance

Table 5: GRU RMSE Relative Performance

Table 6: *

Relative baseline RMSE performance compared to GRU a averaged over all countries: lower is better GRU performance

Time (Q's Ahead)	1Q	2Q	3Q	4Q	5Q
Country-by-Country VAR(1)	5.1	5.3	5.5	5.4	5.5
Country-by-Country AR(2)	5.0	5.0	5.2	5.2	5.3
Single Model GRU	**4.8**	**4.9**	**5.0**	**5.2**	**5.2**

Model Bias and Variance

Table 7:

	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.

Conclusion

- Our neural network forecasting model was competitive with the best economic forecasting models
- As neural networks have many architecture choices, we plan to add even more state-of-the-art improvements
- With the suggestive evidence of policy invariance, in future work, we plan use these models for counterfactual analysis

Thank You

camfen@umich.edu

Appendix

Dense Layers

- A dense layer in a neural network is simply a vector regression $y = \sigma(\beta x)$ with a link function (called an activation), σ, which makes the model nonlinear
- y is a vector and so β is a matrix
- y becomes the multivalued input of the next layer, ie

$$y_3 = \sigma(\beta_3(y_2)) = \sigma(\beta_3(\sigma(\beta_2 x))) \tag{4}$$

- The activations σ are essential because a linear combination of linear transformations is still linear so without the activations the model doesn't become more expressive

Dense Layers: An Example

- Our input, x, is a 3 dimensional vector (GDP, consumption, unemployment) with 250 time-steps
- Like in logistic regression, x is input into the first layer: $y_2 = \sigma(\beta_2 x)$
- y_2 is now the input, like x, into the second layer
  - If we want y_2 to be size 128, then by definition β_2 is 128 x 3
- If,
$$y_3 = \sigma(\beta_3 y_2) = \sigma(\beta_3(\sigma(\beta_2 x))) \tag{5}$$
  and y_3 is the 1 dimensional output, then since y_2 is 128 x 250 then β_3 is 1 x 128
- In this case, y_2 is the only hidden layer, but you can imagine having many more hidden layers before producing an output