Economic Nowcasting with Long Short-Term Memory Artificial Neural Networks (LSTM)

Daniel Hopp

Artificial neural networks (ANNs) have been the catalyst to numerous advances in a variety of fields and disciplines in recent years. Their impact on economics, however, has been comparatively muted. One type of ANN, the long short-term memory network (LSTM), is particularly well-suited to deal with economic time-series. Here, the architecture’s performance and characteristics are evaluated in comparison with the dynamic factor model (DFM), currently a popular choice in the field of economic nowcasting. LSTMs are found to produce superior results to DFMs in the nowcasting of three separate variables; global merchandise export values and volumes, and global services exports. Further advantages include their ability to handle large numbers of input features in a variety of time frequencies. A disadvantage is the stochastic nature of outputs, common to all ANNs. In order to facilitate continued applied research of the methodology by avoiding the need for any knowledge of deep-learning libraries, an accompanying Python (Hopp 2021a) library was developed using PyTorch. The library is also available in R, MATLAB, and Julia.

Key words: Forecasting; machine learning; python.

1. Introduction

A defining feature of the 21st century so far has been the explosion in both the volumes and varieties of data generated and stored (Domo 2017). Almost every industry and aspect of life has been affected by this “data revolution” (Einav and Levin 2014; MacFeely 2020). Simultaneously, rapid advancements in machine learning methods have been made, spurred on in part by the need for novel methods to analyze these new data quantities. Perhaps no methodology has gained greater prominence than the artificial neural network (ANN). ANNs are the engine behind tremendous leaps in fields as disparate as machine translation, image recognition, recommendation engines, and even self-driving vehicles. Yet to date, their impact in the field of economic policy has been largely muted or exploratory in nature (Falat and Pancikova 2015).

This is not to suggest that economic data have been immune to the transformative forces of the data revolution. Quite the opposite in fact, as classical economic data series from national statistical offices (NSO) and other organizations can now be fortified by alternative data sources like never before, helping to provide glimpses into the developments of the global economy with unparalleled granularity and timeliness. The
COVID-19 pandemic and ensuing economic crisis showcased this, with analysts and policy-makers gaining insight to the rapidly evolving economic situation from such alternative data sources as Google mobility data (Yilmazkuday 2021), booking information from dining apps (OpenTable 2021), and transaction data from e-commerce sites (Statista 2021), among many others.

The availability of a broad range of novel, timely indicators should ostensibly have led to significant advances in the field of economic nowcasting, where real-time macroeconomic variables that may be published with a significant lag are estimated based on an array of more timely indicators (Banbura et al. 2010; Giannone et al. 2008). In reality, the field has not experienced the degree of progress seen in other fields, such as image recognition, in the past ten years. A large factor in this relative stagnation is the fact that many of the issues facing nowcasting are not addressed by more data alone. Issues such as multicollinearity, missing data, mixed-frequency data, and varying publication dates are sometimes even exacerbated by the addition of variables (Porshakov et al. 2016). As such, advancements in the field come from a combination of both new data and methodological developments. Dynamic factor models (DFM) in particular have been found to address many of the data issues inherent in nowcasting (Stock and Watson 2002), and have been applied successfully in applications such as nowcasting economic growth in 32 countries (Matheson 2011), nowcasting German economic activity (Marcellino and Schumacher 2010), and nowcasting Canadian GDP growth (Chernis and Sekkel 2017).

The basic premise of DFMs is that one or more latent factors dictates the movement of many different variables, each with an idiosyncratic component in relation to the factor(s). With historical data, the factor(s) can be estimated from the variables. Subsequently, even in future periods where not all data are complete, the factor(s) can still be estimated and used to generate forecasts for variables that are not yet published, as each variable’s relation to the factor(s) has already been estimated.

Despite DFMs’ strengths in addressing a wide swath of nowcasting’s data issues, the impressive performance of ANNs in other domains raises the question of their performance in nowcasting. ANNs have been applied to economic nowcasting in the past (Loermann and Maas 2019). However, due to the time-series nature of many economic nowcasting applications, the long short-term memory (LSTM) architecture is better suited to the problem than the traditional feedforward architecture explored in Loermann and Maas (2019). LSTMs are an extension of recurrent neural network (RNN) architecture, which introduces a temporal component to ANNs. LSTMs have been used to nowcast meteorological events (Shi et al. 2015) as well as GDP (Kurihara and Fukushima 2019).

However, use of LSTMs in nowcasting economic variables remains in its infancy, perhaps partly due to high barriers to their implementation. Many common deep learning frameworks, including Keras and PyTorch, include provisions for LSTMs. However, the implementations are general and require knowledge of the frameworks to successfully implement. As such, a Python library focused on economic nowcasting has been published alongside this article, available for install on PyPi (Hopp 2021a) along with wrappers for R, MATLAB, and Julia. Hopefully, a more accessible library will help stimulate interest and expand the applications of these models. More information on the libraries is available at:
The remainder of this article is structured as follows: the next section will further explain nowcasting and its challenges; Section 3 will explore ANNs and LSTMs in more detail; Section 4 will examine the LSTM’s empirical performance compared with DFMs in nowcasting three series: global merchandise trade exports expressed in both values and volumes and global services exports; the final section will conclude and examine areas of future research.

2. Exposition of Nowcasting Problem

Nowcasting, a portmanteau of “now” and “forecast”, is the estimation of the current, or near to it either forwards or backwards in time, state of a target variable using information that is available in a timelier manner. Keith Browning coined the term in 1981 (WMO 2017) to describe forecasting the weather in the very near future based on its current state. The concept and term remained in the meteorological domain for years before being adopted into the economic literature in the 2000s. The concept of real-time estimates of the macroeconomic situation predates the adoption of the nowcasting terminology, as evidenced by Mariano and Murasawa (2003). However, Giannone et al. (2005) explicitly referenced the term “nowcasting” in its title and the term became commonplace in subsequent years, being applied for example to Portuguese GDP in 2007 (Morgado et al. 2007) and to Euro area economic activity in 2009 (Giannone et al. 2009). The 2010s saw a wealth of papers examining the topic both for a range of target variables as well as with a range of methodologies. Targets most often included GDP (Rossiter 2010; Bok et al. 2018), and trade (Cantú 2018; Guichard and Rusticelli 2011). Common methodologies include dynamic factor models (DFM) (Guichard and Rusticelli 2011; Antolin-Diaz et al. 2020), mixed data sampling (MIDAS) (Kuzin et al. 2009; Marcellino and Schumacher 2010) and mixed-frequency vector autoregression (VAR) (Kuzin et al. 2009), among others. Nowcasting also has relevance in the context of the 2030 Agenda for Sustainable Development (UN 2015). Many indicators face issues in terms of data quality, availability, timeliness, or all three. As such, nowcasting is being discussed as a possible method of ensuring maximum coverage in terms of indicators (UNSD 2020).

Economic nowcasting is generally confronted with three main issues regarding data. The first is mixed frequency data, or when all independent variables and the dependent variable are not recorded with the same periodicity. This occurs frequently in economic data, for instance when trying to nowcast a quarterly target variable, such as GDP growth, using monthly indicators, or estimating a yearly target variable with a mixture of monthly and quarterly variables. The second is the heterogeneous publication schedules of independent variables, frequently referred to as “ragged edges”. Any nowcasting methodology must provide provisions for incomplete or partially complete data, as varying availability of latest data is the reality of most data sets of economic series. Finally, there is the issue of the “curse of dimensionality”, which renders many classical econometric methods less effective in the nowcasting context and hinders the application.
of “big data” to the field (Buono et al. 2017). The problem stems from the nature of many economic variables, where they may have few observations relative to the potential pool of explanatory variables or features. The quarterly target series for the United Nations Conference on Trade and Development’s (UNCTAD) nowcasts for global merchandise trade, for instance, only began in 2005 (Cantú 2018). That leaves only 60 observations for training a model at the end of 2020. Meanwhile, many more than 60 potential independent variables can be conceived of to estimate a model of global merchandise trade.

The nowcasting methodologies previously mentioned address these problems in varying ways to achieve better predictions, and LSTMs are no different. The following section will provide background information on their network architecture as well as the characteristics that allow them to address the aforementioned nowcasting data problems. Those interested in an even more comprehensive examination of neural networks should see Gurney (1997).

3. Artificial Neural Networks and Long Short-Term Memory Networks

3.1. Artificial Neural Networks and Recurrent Neural Networks

3.1.1. Layers, Nodes, and Weights

ANNs are made up of various inter-connected layers composed of groups of nodes or neurons. Figure 1 shows one of the simplest forms of neural network, a dense, single layer

Fig. 1. A single layer feedforward neural network.
feedforward network. Feedforward means that signal or information flows only one way through the network, from input, then through any intermediate layers, finally to output. Dense, or fully connected, means that each node of every layer is connected to each node of the next layer. This distinction is superfluous in the single layer case, but relevant in more complex network architectures.

This simple network can be interpreted in the following manner: four input variables, represented by the four nodes in the input layer, are multiplied by four coefficients or weights, represented by the solid lines connecting each input node to the output node, then summed to obtain an output or prediction. This sounds similar to linear regression because, in this simplified case and in the absence of an activation function, which will be discussed below, it essentially is. The output layer can have more than one node, for instance in cases of categorical classification, but only the single output layer node case is relevant for the regression application in this article.

Figure 2 shows a more complex network architecture, where one hidden layer is added with two additional nodes. Now, coefficients exist between both the initial input layer and the hidden layer as well as between the hidden layer and the final output layer. Usually, no good semantic interpretation of hidden layers exists as they become an abstracted amalgamation of previous layers. They are best thought of as intermediate processing layers which help the network approximate the target function. However, post analysis on a trained network can sometimes lead to human-interpretable meanings, identifying

![Image](image.png)

**Fig. 2.** A multi-layer feedforward neural network with one hidden layer
hidden layers or neurons associated with the identification of say eyes or textures in a photograph, or with macro-concepts like investment or developing economy performance in an economic context. Those interested in learning more about the interpretation of how and what neural networks learn should see Subsection 10.1 of Molnar (2019). The formula for an individual node on the hidden layer is below.

\[ H_j = \sum_{i=1}^{n} w_i v_i \]  \hspace{1cm} (1)

where:

- \( H_j \) = the value of hidden node \( j \)
- \( n \) = the number of nodes in the previous layer connected to the hidden node
- \( w \) = the weight or coefficient between the previous layer’s node and the hidden node
- \( v \) = the value or output of the previous layer’s node

3.1.2. Activation Functions

The simple networks described above contain a rather large drawback, the fact that they can only represent linear relationships. The ability to approximate complex, non-linear relationships is one of the defining characteristics of ANNs and an essential component of their predictive power. This component is introduced by means of a non-linear activation function. While there are many different types of activation function, see Sharma et al. (2020) for an in depth explanation of several, a commonly used one is the Rectified Linear Unit, ReLU for short. It will be used here as the illustrative case of an activation function. The formula for ReLU is below.

\[ f(x) = \max(0, x) \]  \hspace{1cm} (2)

In words, given an input, if the value is greater than or equal to zero, leave it unchanged, if the value is less than zero, output zero.

To introduce the activation function to our network, we run the result of our weighted sums at each node through the activation function before passing the result on to the next layer.

\[ H_j = K(\sum_{i=1}^{n} w_i v_i) \]  \hspace{1cm} (3)

Equation (3) is the same as Equation (1), with the addition of \( K \), the activation function. A different activation function is typically used for the final output layer, depending on the application. For instance, a sigmoid function may be used for a binary outcome problem, or none at all for a regression problem. The intuition is the following, if we employed the ReLU activation function in the output layer of our network, we would never be able to predict values less than zero with our network.

3.1.3. Backpropagation and Gradient Descent

With a basic understanding of neural network architecture, the next question becomes how this network can learn or be trained to improve the qualities of its predictions. We have
enough information to understand the first step in this process. For the first training epoch, or run of data through the network, all weights in the network are randomly initialized, the input data is fed through, and a prediction is obtained. This initial randomization step is in fact a primary source of ANNs’ stochasticity. Ten identical ANNs trained on the same data will output ten slightly different predictions because their starting points, or initial weights, were all different. This initial prediction will most likely be of poor quality, since weights were chosen randomly. For the model to learn, that is, update its weights, and increase the quality of its predictions, two more steps are necessary: backpropagation and gradient descent.

Although other methods of training a neural network exist, the combination of backpropagation and gradient descent has been by far the most common since the methodology’s introduction for use in neural networks in 1986 (Rumelhart et al. 1986). Before a network can be trained to minimize error or loss, a loss function must be chosen to determine that error. The type of loss function chosen depends on the application. For instance, whether the network is used for classification or regression. Common loss functions for regression problems include mean absolute error (MAE) and mean squared error (MSE). See PyTorch (2021b) for more examples of loss functions. The formula for the MAE loss function is below.

\[
MAE = \frac{1}{n} \sum_{i=1}^{n} |y_i - x_i|
\]  
(4)

where:

- **MAE** = mean absolute error
- **n** = the number of training observations
- **y** = the actual value of an observation
- **x** = the network’s predicted value of an observation

Backpropagation is in turn an algorithm that computes the gradients, or partial derivatives, of this loss function with respect to the weights of each layer in the network using the chain rule. Figure 3 helps illustrate the intuition of this process. Figure 3 shows the plot of a loss function in a network with a single weight. The goal is to minimize loss. In this simple case, we could set our derivative equal to zero and solve for weight. However, in more complicated networks, the loss function does not have a closed form derivative, so gradient descent is employed instead to reduce loss. At our initial weight, the gradient is calculated via the backpropagation algorithm, this information is then used to determine which direction to move the weight to reduce loss, represented by the cluster of arrows pointing towards the function minimum in Figure 3. The process is then repeated. Backpropagation is a rich mathematical field in its own right, interested readers should see Chapter 2 of Nielsen (2015) for a deeper examination.

Backpropagation is only the step that calculates gradients. To actually update weights and thus train the model, gradient descent is then employed. A representation of gradient descent is detailed below.

\[
w_i = w_{i-1} - \delta \nabla C(w_{i-1})
\]  
(5)

where:
In words, a particular weight in the network is equal to its weight in the previous epoch minus the gradient of the cost function at the previous weight times the learning rate. Learning rate, or step size, is an important hyperparameter in ANNs. It dictates how much to update weights by in each training epoch. A very large learning rate risks unstable weights, as the cost function minimum may be continually jumped over. A small learning rate risks moving towards the cost function minimum very slowly. Taking the example of Figure 4, a high learning rate risks jumping to the left and right side of the parabola, never actually reducing loss, while a low learning rate risks moving the weight only slightly down to the left, even after many training epochs. In practice, calculating gradients for all observations, called gradient descent or batch gradient descent, can be computationally expensive, time consuming, and lead to overfitting (Keskar et al. 2017). Accordingly, stochastic gradient descent or mini batch gradient descent is often employed to speed up this process. In stochastic gradient descent, rather than calculating gradients for all observations, they are only calculated for one observation. Mini batch splits the difference between these two approaches, calculating gradients for a subset of observations. It should also be mentioned that networks may not always converge, or reduce loss, during training. This may be due to poor-quality input data, ill-suited hyperparameters, network architecture, choice of activation function, or other factors.

With that, we now understand the basics of how feedforward ANNs are structured and trained. A network architecture is specified with input data connected to an output via intermediary hidden layer(s). The weights between these layers and nodes are initially randomized. Data is passed through these weights and layers, run through a non-linear activation function, and an output is obtained. This output is then run through a cost function, which is then used to calculate gradients via backpropagation. Gradient descent is then employed to update the network’s weights in the direction of reducing loss. The
input data is then run through the network again, and the process is repeated until loss no longer decreases or after a predetermined number of training epochs has been reached. Weights are then fixed at final values and the network is considered trained. New data can then be fed through to obtain new outputs or predictions.

3.1.4. Recurrent Neural Networks

Feedforward networks are extremely powerful and have proven very effective in a variety of applications. However, as their name suggests, the flow of information through the network is unidirectional. This limits their usefulness in applications with a temporal aspect. They can, however, still be used with time series by flattening the data, that is, converting each lag of a variable into a separate column. In some cases, this approach can produce good predictions and can be considered before turning to the more explicitly time-based architectures detailed below (Brownlee 2018). Recurrent neural networks address feedforward networks’ temporal deficiency by introducing feedback loops and converting the network into a directed acyclic graph (Amidi and Amidi 2019; Stratos 2020; Dematos et al. 1996). Figure 4 illustrates a simple RNN.

Fig. 4. A recurrent neural network.

The network is similar to the feedforward network in Figure 2, but the outputs of the hidden layer are fed back into the network rather than directly to the output layer. The introduction of the temporal component means that the entirety of the network can no
longer be fully expressed in a diagram like Figure 4. Rather, each hidden node now conceals more folded in layers. To better illustrate this concept, Figure 5 displays one of the hidden nodes from Figure 4 unfolded.

where:

\[ x = \text{input from the previous layer to the hidden node} \]
\[ s = \text{state of the hidden node} \]
\[ o = \text{output of the hidden node} \]
\[ u = \text{weight between the input layer and the hidden layer} \]
\[ w = \text{weight between the previous hidden state of the node and the current state} \]
\[ v = \text{weight between the hidden layer and the output layer} \]

The mathematics for the cost function, backpropagation, and gradient descent remain the same, with the additional dimension of time. This temporal component makes RNNs well-suited for applications such as natural language processing or speech processing. However, due to vanishing or exploding gradients, which give RNNs a short memory, their usefulness in nowcasting is limited (Grosse 2017).

Vanishing and exploding gradients arise from the mechanics of backpropagation through time. With the introduction of time, the cost function is now calculated at each point in time, so weights need to be updated not only for each node, but for each node at each point in time. Additionally, later nodes are dependent on the input from earlier nodes, so weights \( w \) from Figure 5 are multiplied many times throughout the network and can thus tend towards zero or extremely large numbers. Because of temporal dependencies where later weights are based on early weights, this is an issue for the entire network. Long short-term memory networks, discussed in the next section, are one way of addressing this issue.

Fig. 5. Unfolded recurrent neural network node.
3.2. Long Short-Term Memory Networks

3.2.1. LSTM Architecture

Long short-term memory networks (LSTM) introduce three gates, a forget gate, an input gate, and an output gate, to RNN nodes (Chung et al. 2014). Crucially, these gates allow gradients to flow unchanged through the network, mitigating the exploding and vanishing gradients problem. Figure 6 displays an LSTM node with time $t$ hidden node enlarged for detail. The graphical representation of an LSTM node is rather complicated, if readers still have trouble grasping the concepts, see Olah (2015).

where:

- $x$ = input from the previous layer to the hidden node
- $s$ = state of the hidden node
- $o$ = output of the hidden node
- $u$ = weight between the input layer and the hidden layer
- $w$ = weight between the previous hidden state of the node and the current state
- $v$ = weight between the hidden layer and the output layer
- $c$ = memory state
- $\sigma$ = sigmoid layer
- $\tanh$ = tanh layer

In contrast with the RNN diagram of Figure 5, we see that the LSTM node now has three inputs: data from the previous layer, $x$, the previous state or output of the node, $s_{t-1}$, and $c$, the memory state of the previous node. $c$ is what allows gradients to flow through the network and gives LSTMs a longer memory. There are essentially three things happening

Fig. 6. Detail of an LSTM node.
in an LSTM node: first, deciding what information to keep or discard in the memory state; second, deciding what new information to introduce into the memory state; third, deciding what to output to the next layer.

These three steps are represented in Figure 6 by the three horizontal lines leading into and out of $c$ in the hidden node at time $t$. Moving sequentially from top to bottom, the first horizontal line is the forget gate. The $\sigma$ on this line represents a layer with the sigmoid activation function, which outputs a value between zero and one. This represents how much information from the previous memory state to allow to pass onwards. The next step, choosing what to introduce to the memory state, has two parts. First, simultaneously another sigmoid layer, the input gate, determines which values to update while a tanh layer scales those potential values to add to the memory state. The output of these two layers is then combined to update the memory state. At this point, the memory state is finished updating and is passed on to the next node. However, one last step is carried out to determine the output of the node. That is, the output of the node as we understand it from feedforward networks and RNNs, for instance to the final output layer. In a process similar to the second step, a sigmoid layer determines which parts of the memory state to output, while a tanh layer transforms the values of the memory state. These two are then combined to determine the output of the node.

3.2.2. LSTM Suitability for Nowcasting

LSTMs’ ability to address the first common nowcasting data issue, mixed frequency data, stems from ANNs’ ability to learn complex, non-linear relationships in data, a product of multiple neuron layers coupled with non-linear activation functions. As such, mixed frequency data can be fed to the network in the highest frequency available, with lower frequency data having missings at time periods where data are not published. These missing data can then be filled using a variety of approaches, including with the mean, the median, with values sampled from a distribution (Ennett et al. 2001), or with other more complex methods (Smieja et al. 2019). In the analysis performed in this article, mean replacement was chosen and implemented in the accompanying Python library.

LSTMs are able to address the ragged edges problem through no special mechanism other than standard missing-filling methods. These include using ARMA or VAR models to fill in ragged-edges (Kozlov et al. 2018), as well as using the mean or Kalman filters (Doz et al. 2011). The method chosen in the context of LSTM nowcasting can be considered a hyperparameter to be tuned and tested empirically. At the time of writing, the Python library supports ARMA filling and any n-to-1 series transformation, for example, mean, median, etc. Both ARMA and mean filling were used in the analysis performed in this article, depending on the results of hyperparameter tuning for each individual model.

The last major problem of nowcasting, the curse of dimensionality, is partially addressed by LSTMs’ efficiency compared with other methods, i.e., their computation time scales very slowly with the number of variables (Hochreiter and Schmidhuber 1997). In empirical testing, the DFM’s computation time scales exponentially with the number of features, while the LSTM’s remains constant. Both methodologies’ computation time scales linearly with the number of observations.

As a result of this efficiency, a functional model can be trained with many more features than a DFM. Computational feasibility of models with a large number of input variables is
However only part of how LSTMs and ANNs address the curse of dimensionality. Neural networks are remarkable for their ability to extract relevant features from higher dimensional spaces and project them onto lower dimensional spaces, which is already a form of feature reduction (Hodas and Stinis 2018), and for their robustness to multicollinearity (De Veaux and Ungar 1994). This is why incredibly complex networks with millions or more coefficients can still generate effective predictions with many fewer training observations. They are also compatible with standard dimensionality and over-fitting reduction techniques, such as regularization. For instance, L2 regularization can be introduced to the nowcast_lstm library via the PyTorch optimizer function’s weight_decay argument. PyTorch’s dropout parameter is another effective means of regularization implemented in the nowcast_lstm library (PyTorch 2021a). Within the LSTM architecture, as in any ANN, there are many choices to be made regarding network architecture and hyperparameters. Some examples include the number of hidden states, the number of layers, the loss function and the activation function, among many others.

4. Empirical Analysis

4.1. Description of Data and Models

In order to assess the relative performance of LSTMs vs DFMs, three target variables were used: global merchandise exports in both value (WTO 2020) and volume (UNCTAD 2020a), and global services trade (UNCTAD 2020a). These are the same series UNCTAD currently produces nowcasts for using DFMs (UNCTAD 2020b) and which were examined in an UNCTAD research paper (Cantu 2018). The target series are all quarterly. A large pool of 116 mixed-frequency monthly and quarterly independent series was used to estimate each of the target series. These series are listed in Online Supplemental Material while more information on any individual series is available upon request. All series were converted to seasonally adjusted growth rates using the US Census Bureau’s X13-ARIMA-SEATS methodology (USCB 2017). Modelling on seasonally adjusted data is standard practice in nowcasting and has been shown to produce empirically better results with DFMs than using non-seasonally adjusted data (Camacho et al. 2015).

The DFM model used was the same examined in Cantú (2018). In this model, the DFM is modeled in a state-space representation where it is assumed that the target and independent variables share a common factor as well as individual idiosyncratic components. The Kalman filter is then applied, and maximum likelihood estimates of the parameters obtained. This is a common method of estimating DFMs and is explained in further detail in Banbura and Rünstler (2011). The LSTM model used was that present in the nowcast_lstm Python library using the average of 10 networks’ output with basic hyperparameter tuning. Table 1 lists all hyperparameters and values used for tuning. The logic of averaging the output of more than one network to obtain predictions is discussed further in Subsection 4.4, but see Stock and Watson (2004) for a discussion of forecast combination. There exist different variants of LSTMs, but PyTorch’s LSTM class, which is a classic or general LSTM architecture, served as the base for the library (PyTorch 2021c), with a general structure of a variable number of LSTM layers plus one linear densely connected layer for generating final predictions.
4.2. Modelling Steps

Hyperparameter tuning of the LSTM and model performance was evaluated using a training set dating from the second quarter of 2005 to the third quarter of 2016 and a test set dating from the fourth quarter of 2016 to the fourth quarter of 2019. Time series are generally not suitable for cross-fold validation, a model validation technique often used to avoid overfitting. Cross validation involves the random selection of subsamples from the data to train and assess models on different data. For more information on the topic, see Scikit-learn (2021). Time series, however, are often not independent and it makes little intuitive sense to predict the past using information from the future. Though there do exist methods of using cross validation with time series, for instance using rolling points in time in the past as the folds, they were not used in this analysis due to the already relatively short time span of the data.

A pool of independent variables was used to ensure the robustness of results, as either model could perform better on a single set of features due to chance. As such, the models’ performance was evaluated by taking random samples of between five and 20 features, then fitting both an LSTM and DFM model on this same sample. Both methods’ performance was then evaluated on the test set via mean absolute error (MAE) and root-mean-square error (RMSE) on five different data vintages, repeating the process 100 times for each of the three target variables. In this manner, a distribution of relative performance over a wide breadth of independent variables could be obtained. The number of features was restricted to a maximum of 20 due to the high computational time of estimating DFMs with more than this number, though performance of the LSTM may have been improved further if additional features were used.

Data vintages in this case refer to the artificial withholding of data to simulate what the availability of data would have looked like at different points in the past. This is important in evaluating model performance in the nowcasting context, as in real life series have varying publication schedules which nowcasting models must be robust to. The five vintages simulated were: two months before the target period, for example, if the target was the second quarter of 2019, the data as it would have appeared in April 2019; one month before; the month of; a month afterwards; and two months afterwards. The model continues to be evaluated even after the target period has theoretically “passed” as data continue to be published for a given month well after it has passed, depending on the series’ individual publication schedule. For example, two months after the second quarter of 2019 simulates being in August 2019, when much more data on the second quarter is

| Hyperparameter               | Values                      |
|------------------------------|----------------------------|
| Batch Size                   | 15, 30, 60                 |
| Loss function                | L1, MSE                    |
| Number of hidden layers      | 10, 20, 40                 |
| Number of LSTM layers        | 2, 4                       |
| Number of training epochs    | 50, 100, 200               |
| Ragged edge filling method   | mean, ARMA                 |

Table 1. Hyperparameters and values used for tuning.
available. The variables’ publication lags were obtained based on empirical observations from the period from April to November 2020.

4.3. Relative Performance

Figure 7 shows the distribution of the LSTM’s error as a proportion of the DFM’s for each target variable. A value less than one for an individual model indicates better performance on the test set for the LSTM, while a value greater than one indicates worse performance. Consequently, a distribution centered around one, that is, the vertical line, indicates

![LSTM error as a proportion of DFM error](image)

Fig. 7. LSTM error as a proportion of DFM error.
comparable performance between the two models, while one to the left of the vertical line indicates better performance on average for the LSTM model.

The results clearly favor the LSTM model, obtaining better average performance for both performance metrics across all data vintages and target variables, with the sole exception of RMSE for the two months before services exports vintage. Tables 2, 3 and 4 display the average performance metrics for the two models over the sample of 100 different feature combinations, as well as the results using a simple autoregressive model as a benchmark. A one-tailed t-test was performed on the LSTM and DFM errors to ascertain the significance of these differences in performance, with the alternative hypothesis that the LSTM errors were lower. Results are displayed in the LSTM columns.

Table 2. Average performance metrics, global merchandise trade exports, values.

| Vintage            | ARMA MAE | LSTM MAE | DFM MAE | ARMA RMSE | LSTM RMSE | DFM RMSE |
|--------------------|----------|----------|---------|-----------|-----------|----------|
| 2 months before    | 0.0177   | 0.0149   | 0.0150  | 0.0233    | 0.0176**  | 0.0185   |
| 1 month before     | 0.0177   | 0.0112*  | 0.0117  | 0.0233    | 0.014     | 0.0141   |
| month of           | 0.0177   | 0.0115   | 0.0118  | 0.0233    | 0.0142    | 0.0142   |
| 1 month after      | 0.0168   | 0.0108***| 0.0117  | 0.0217    | 0.0138    | 0.0142   |
| 2 months after     | 0.0168   | 0.0094***| 0.0109  | 0.0217    | 0.0119*** | 0.0135   |
| Average            | 0.0173   | 0.0115** | 0.0122  | 0.0227    | 0.0143    | 0.0149   |

Note: *p < .05 **p < .01 ***p < .001.

Table 3. Average performance metrics, global merchandise trade exports, volumes.

| Vintage            | ARMA MAE | LSTM MAE | DFM MAE | ARMA RMSE | LSTM RMSE | DFM RMSE |
|--------------------|----------|----------|---------|-----------|-----------|----------|
| 2 months before    | 0.0085   | 0.006**  | 0.0064  | 0.0097    | 0.0075**  | 0.0078   |
| 1 month before     | 0.0085   | 0.0051***| 0.0066  | 0.0097    | 0.0066*** | 0.0079   |
| month of           | 0.0085   | 0.0049***| 0.0065  | 0.0097    | 0.0063*** | 0.0079   |
| 1 month after      | 0.0084   | 0.0045***| 0.0057  | 0.0108    | 0.0059*** | 0.0069   |
| 2 months after     | 0.0084   | 0.0042***| 0.0056  | 0.0108    | 0.0054*** | 0.0067   |
| Average            | 0.0085   | 0.0049***| 0.0062  | 0.0101    | 0.0063*** | 0.0074   |

Note: *p < .05 **p < .01 ***p < .001.

Table 4. Average performance metrics, global services exports.

| Vintage            | ARMA MAE | LSTM MAE | DFM MAE | ARMA RMSE | LSTM RMSE | DFM RMSE |
|--------------------|----------|----------|---------|-----------|-----------|----------|
| 2 months before    | 0.0119   | 0.0123***| 0.0129  | 0.0151    | 0.0154    | 0.0152   |
| 1 month before     | 0.0119   | 0.0103***| 0.0113  | 0.0151    | 0.0135**  | 0.0140   |
| month of           | 0.0119   | 0.0103***| 0.0111  | 0.0151    | 0.0135**  | 0.0141   |
| 1 month after      | 0.0119   | 0.0103***| 0.0115  | 0.0151    | 0.0137*** | 0.0146   |
| 2 months after     | 0.0119   | 0.0101***| 0.0117  | 0.0151    | 0.0134*** | 0.0147   |
| Average            | 0.0119   | 0.0107***| 0.0117  | 0.0151    | 0.0139*** | 0.0145   |

Note: *p < .05 **p < .01 ***p < .001.
An additional metric for comparing the performance specifically of two forecasts is the Diebold-Mariano test. Table 5 displays the proportion of models where a one-sided Diebold-Mariano test was significant at the 5% level. The “DFM” column displays results where the alternative hypothesis was that the DFM was a more accurate forecast than the LSTM, while the “LSTM” column displays the reverse. The results closely mirror the findings in Figure 7 and Tables 2–4. In all target-vintage combinations but three the LSTM had a higher proportion of significance.

Tables 6 and 7 display additional information on the two methodologies’ errors on the test set, namely average skewness, kurtosis, and various quantiles of absolute errors.

### Table 5. Proportion of models with forecasts on the test set better than those of the other methodology, according to one-sided Diebold-Mariano test at 5% significance level.

| Target     | Vintage       | DFM | LSTM |
|------------|---------------|-----|------|
| Values     | 2 months before | 4%  | 10%  |
| Values     | 1 month before | 9%  | 6%   |
| Values     | month of       | 2%  | 7%   |
| Values     | 1 month after  | 2%  | 9%   |
| Values     | 2 months after | 1%  | 17%  |
| Volumes    | 2 months before | 0%  | 3%   |
| Volumes    | 1 month before | 0%  | 9%   |
| Volumes    | month of       | 0%  | 15%  |
| Volumes    | 1 month after  | 1%  | 15%  |
| Volumes    | 2 months after | 1%  | 26%  |
| Services   | 2 months before | 6%  | 2%   |
| Services   | 1 month after  | 1%  | 0%   |
| Services   | month of       | 0%  | 4%   |
| Services   | 1 month after  | 0%  | 10%  |
| Services   | 2 months after | 0%  | 21%  |

Note: For brevity, “Values” refers to global merchandise exports in values, “Volumes” refers to global merchandise exports in volumes, and “Services” refers to global services exports.

### Table 6. DFM, average skewness, kurtosis, and quantiles of absolute errors on the test set.

| Target     | Vintage        | Skewness | Kurtosis | 50th quantile | 75th quantile | 90th quantile | 95th quantile | 99th quantile |
|------------|----------------|----------|----------|----------------|----------------|----------------|----------------|----------------|
| Values     | 2 month before | -0.198   | 2.437    | 0.013          | 0.021          | 0.028          | 0.033          | 0.037          |
| Values     | 1 month before | 0.078    | 2.223    | 0.011          | 0.017          | 0.021          | 0.024          | 0.026          |
| Values     | month of       | 0.113    | 2.268    | 0.011          | 0.016          | 0.022          | 0.025          | 0.027          |
| Values     | 1 month after  | -0.013   | 2.401    | 0.010          | 0.016          | 0.022          | 0.026          | 0.028          |
| Values     | 2 months after | 0.168    | 2.644    | 0.015          | 0.021          | 0.024          | 0.026          | 0.028          |
| Volumes    | 2 months before | -0.310  | 2.544    | 0.009          | 0.012          | 0.014          | 0.015          | 0.015          |
| Volumes    | 1 month before | -0.205   | 2.391    | 0.009          | 0.012          | 0.014          | 0.015          | 0.015          |
| Volumes    | month of       | -0.123   | 2.318    | 0.009          | 0.012          | 0.014          | 0.015          | 0.015          |
| Volumes    | 1 month after  | -0.077   | 2.392    | 0.005          | 0.011          | 0.012          | 0.013          | 0.013          |
| Volumes    | 2 months after | 0.118    | 2.114    | 0.005          | 0.008          | 0.010          | 0.011          | 0.012          |
| Services   | 2 months before | 0.634   | 2.476    | 0.012          | 0.017          | 0.022          | 0.025          | 0.029          |
| Services   | 1 month before | 0.394    | 2.992    | 0.010          | 0.015          | 0.022          | 0.025          | 0.029          |
| Services   | month of       | 0.715    | 3.222    | 0.009          | 0.015          | 0.023          | 0.026          | 0.030          |
| Services   | 1 month after  | 0.676    | 3.314    | 0.009          | 0.015          | 0.024          | 0.028          | 0.031          |
| Services   | 2 months after | 0.713    | 3.170    | 0.009          | 0.016          | 0.023          | 0.027          | 0.031          |

Note: For brevity, “Values” refers to global merchandise exports in values, “Volumes” refers to global merchandise exports in volumes, and “Services” refers to global services exports.
To test for autocorrelation of errors, a portmanteau test was performed on the test predictions for all models (Johansen 1995). Average p-values are presented in Table 8, with the null hypothesis that autocorrelation was not present in the residuals. At the individual model level, only 11 were able to reject the null hypothesis at a 5% significance level, all DFMs predicting global merchandise exports values at various time vintages. To test for heteroskedasticity in the residuals, a Lagrange Multiplier test was additionally performed, with the null hypothesis that the residuals were homoskedastic (Engle 1982). Average p-values are presented in Table 9. No individual models were able to reject the null hypothesis at a 5% significance level. The results of these tests imply that neither the DFM nor the LSTM are likely to suffer from systematic issues of autocorrelation or heteroskedasticity in their errors. Table 8 displays average bias and variance of the two models at different vintages. Bias and variance were calculated individually for each model in the sample then averaged by target variable-data vintage combination (Table 10).

In terms of bias, the two methods are comparable, with relative performance varying depending on the target series. Broadly, the LSTM had a lower bias on average for

| Target   | Vintage         | Skewness | Kurtosis | 50th quantile | 75th quantile | 90th quantile | 99th quantile |
|----------|-----------------|----------|----------|----------------|----------------|----------------|----------------|
| Values   | 2 months before | −0.247   | 1.820    | 0.014          | 0.021          | 0.026          | 0.028          | 0.030          |
| Values   | 1 month before  | −0.083   | 2.481    | 0.009          | 0.016          | 0.022          | 0.024          | 0.027          |
| Values   | month of        | −0.011   | 2.435    | 0.009          | 0.016          | 0.021          | 0.024          | 0.027          |
| Values   | 1 month after   | −0.224   | 2.555    | 0.008          | 0.016          | 0.021          | 0.024          | 0.027          |
| Values   | 2 months. after | −0.296   | 2.668    | 0.008          | 0.013          | 0.018          | 0.021          | 0.024          |
| Volumes  | 2 months before | −1.036   | 3.635    | 0.005          | 0.008          | 0.010          | 0.013          | 0.016          |
| Volumes  | 1 month before  | −1.024   | 4.052    | 0.004          | 0.007          | 0.009          | 0.012          | 0.015          |
| Volumes  | month of        | −0.850   | 3.633    | 0.004          | 0.007          | 0.009          | 0.011          | 0.014          |
| Volumes  | 1 month after   | −0.763   | 3.638    | 0.004          | 0.006          | 0.008          | 0.011          | 0.013          |
| Volumes  | 2 months after  | −0.728   | 3.611    | 0.003          | 0.005          | 0.007          | 0.010          | 0.012          |
| Services | 2 months before | 0.280    | 2.387    | 0.011          | 0.018          | 0.024          | 0.028          | 0.031          |
| Services | 1 month before  | 0.383    | 2.768    | 0.008          | 0.014          | 0.023          | 0.026          | 0.028          |
| Services | month of        | 0.508    | 2.792    | 0.007          | 0.014          | 0.023          | 0.026          | 0.028          |
| Services | 1 month after   | 0.564    | 2.938    | 0.007          | 0.013          | 0.024          | 0.027          | 0.029          |
| Services | 2 months after  | 0.777    | 3.104    | 0.008          | 0.013          | 0.022          | 0.026          | 0.029          |

Note: For brevity, “Values” refers to global merchandise exports in values, “Volumes” refers to global merchandise exports in volumes, and “Services” refers to global services exports.

Table 8. Average p-values of portmanteau test for autocorrelation.

| Vintage        | DFM, values | LSTM, values | DFM, volumes | LSTM, volumes | DFM, services | LSTM, services |
|----------------|-------------|--------------|--------------|---------------|---------------|---------------|
| 2 months before| 0.72        | 0.69         | 0.70         | 0.92          | 0.82          | 0.71          |
| 1 month before | 0.65        | 0.73         | 0.73         | 0.95          | 0.83          | 0.72          |
| month of       | 0.49        | 0.68         | 0.72         | 0.92          | 0.87          | 0.74          |
| 1 month after  | 0.57        | 0.74         | 0.73         | 0.88          | 0.87          | 0.84          |
| 2 months after | 0.84        | 0.76         | 0.75         | 0.86          | 0.83          | 0.84          |

Note: For brevity, “Values” refers to global merchandise exports in values, “Volumes” refers to global merchandise exports in volumes, and “Services” refers to global services exports.
merchandise exports in terms of values, while the DFM had a lower bias on average for merchandise exports in terms of volumes. For services, which methodology had a lower bias depended on the vintage. Variance in the LSTM was lower than that of the DFM in all target-vintage combinations, though the degree to which this was the case varied. The fact that the LSTM is able to combine higher accuracy with lower volatility suggests the DFM may be overly reactive to signals in the data, predicting large changes in the target variable that are either inaccurate or excessive.

4.4. Comparison With the Dynamic Factor Model

The fact that the LSTM performed better than the DFM on average for all three target variables across almost all vintages and both performance metrics is strong evidence for their relevance in the economic nowcasting space. The LSTM’s ability to handle long-term temporal dependencies due to its architecture may be a source of some of this improved

| Vintage          | DFM, values | LSTM, values | DFM, volumes | LSTM, volumes | DFM, services | LSTM, services |
|------------------|-------------|--------------|--------------|---------------|---------------|---------------|
| 2 months before  | 0.94        | 1.00         | 0.98         | 0.99          | 0.95          | 0.90          |
| 1 month before   | 0.97        | 0.94         | 0.98         | 0.96          | 0.99          | 0.82          |
| month of         | 0.98        | 0.97         | 0.98         | 0.98          | 0.97          | 0.83          |
| 1 month after    | 0.97        | 0.97         | 0.97         | 0.95          | 0.94          | 0.83          |
| 2 months after   | 0.97        | 0.96         | 0.98         | 0.96          | 0.96          | 0.91          |

Note: For brevity, “Values” refers to global merchandise exports in values, “Volumes” refers to global merchandise exports in volumes, and “Services” refers to global services exports.

| Target    | Vintage       | DFM bias  | LSTM bias | DFM variance | LSTM variance |
|-----------|---------------|-----------|-----------|--------------|---------------|
| Values    | 2 months before | -0.004833 | -0.001999 | 0.000265     | 0.000096      |
| Values    | 1 month before | -0.003976 | -0.002097 | 0.000412     | 0.000174      |
| Values    | month of       | -0.004144 | -0.002134 | 0.000444     | 0.000238      |
| Values    | 1 month after  | -0.004955 | -0.003545 | 0.000352     | 0.000260      |
| Values    | 2 months after | -0.003793 | -0.004156 | 0.000415     | 0.000330      |
| Volumes   | 2 months before | 0.000195  | 0.000854  | 0.000060     | 0.000013      |
| Volumes   | 1 month before | 0.000543  | 0.000879  | 0.000096     | 0.000024      |
| Volumes   | month or       | 0.000511  | 0.000684  | 0.000104     | 0.000034      |
| Volumes   | 1 month after  | -0.000018 | 0.000191  | 0.000085     | 0.000039      |
| Volumes   | 2 months after | -0.000043 | 0.000137  | 0.000075     | 0.000044      |
| Services  | 2 months before | -0.001474 | 0.001746  | 0.000070     | 0.000021      |
| Services  | 1 month before | -0.001184 | 0.001913  | 0.000112     | 0.000047      |
| Services  | month of       | -0.001261 | 0.002161  | 0.000116     | 0.000064      |
| Services  | 1 month after  | -0.001729 | 0.001250  | 0.000097     | 0.000073      |
| Services  | 2 months after | -0.001616 | 0.001066  | 0.000105     | 0.000105      |

Note: For brevity, “Values” refers to global merchandise exports in values, “Volumes” refers to global merchandise exports in volumes, and “Services” refers to global services exports.
predictive performance. Of course, the results do not indicate that LSTMs are superior to this type of DFM in every instance. They rather provide some evidence that they can be a competitive alternative to DFMs and have the potential to become a more commonly used methodology in nowcasting. There are, however, characteristics of the methodology with pros and cons relative to DFMs that are independent of predictive performance. One of the pros relative to the DFM was discussed in Section 3. Namely, LSTMs’ ability to handle many more features than the DFM before coming up against computational bottlenecks. This could be beneficial by lessening the need for variable selection in the early stages of an analysis, easing the obtainment of initial results. Additionally, a model is able to be reliably trained on any given set of features and values, which is not the case for the DFM, the training of which may fail if input matrices are non-invertible.

A third advantage is the ability to easily use mixed frequency variables with no corresponding change in the underlying modeling and formulas. Annual, quarterly, monthly, and even theoretically daily data can be combined in a single model just by changing the structure or frequency of the input data, as explained in Subsection 3.2.

Computational speed is more difficult to ascribe to either method as an advantage. There are many factors affecting the computation time of the two models. For DFMs, this includes the number of features, the number of observations, and especially the time taken for maximum likelihood convergence. For LSTMs, this includes the number of observations, as well as nearly all of the hyperparameters. As such, there are cases where either method can be faster. Even still, training a single LSTM network with any choice of hyperparameters is usually faster than estimating a DFM on the same data. For instance, in the 300 model runs of this analysis, this was the case 96% of the time, with the LSTM taking on average just 25% of the time needed to estimate the DFM. However, the results in Figure 7 were obtained by fitting ten LSTM models and averaging the result, in which case the LSTM was faster just 42% of the time, taking on average 2.5 times as long to estimate compared with the DFM. These numbers are slightly skewed in favor of the DFM however, as the number of features was restricted to a maximum of 20. Models with a number of features above this would favor the LSTM in computation time. Because the LSTM needs to be hyperparameter tuned for every new target series, likely more LSTM models need to be estimated than DFM models in the model selection phase of an exercise, leading to comparatively longer run times initially. However, once variables and hyperparameters are selected and models just need to be retrained periodically with the latest data, this disadvantage disappears.

The fact that results were evaluated using ten networks for the LSTM has to do with one of their disadvantages relative to DFMs, namely, the stochastic nature of ANNs. Ten LSTM networks trained on the same data will output ten different predictions due to the randomization of initial weights, which is not the case for DFMs. Training many networks and taking their average predictions is a way to mitigate this characteristic. Figure 8 illustrates how the distribution of predictions develops as more networks are used.

The distributions were obtained by taking a single set of variables predicting global merchandise exports values and training an LSTM model averaging the outputs of between one and 20 networks on the data, then generating a prediction for a single time period. This was repeated 100 times by retraining the same model, thus generating 100 predictions for a single target period, creating the distributions. The predictions were then
While adding more networks can mitigate the stochastic nature of LSTMs’ predictions, adding very many can substantially increase computation time while never achieving perfectly consistent outputs.

A final disadvantage LSTMs have compared with DFMs is the lack of interpretability in their parameters, and the consequent lack of inference as to what is driving changes in the model. DFMs have the advantage of being able to offer precise insights to various features’ impact on predictions, as illustrated by the New York Fed’s nowcasts (Federal Reserve Bank of New York 2021). This is a well-known characteristic of ANNs in general (Fan et al. 2020). In this regard, there is opportunity for further research into applying existing ANN interpretability methods, such as activation maximization or sensitivity analysis (Montavon et al. 2018), to the nowcasting LSTM framework, though one has already been implemented in the `nowcast_lstm` library. The methodology employed is similar to a simplified implementation of calculating Shapley values, whose use in adding interpretability to machine learning methodologies is explained in further detail in

![Distribution of predictions of the same target observation.](image)
5. Conclusion

Timely, accurate estimates of macroeconomic series can be valuable in helping inform policy decisions. This article provides evidence for stronger consideration of LSTMs for this purpose, as well as introduces Python, R, MATLAB, and Julia libraries to facilitate future research. LSTMs were shown to produce superior predictions compared with DFMs on three different target series: global merchandise trade exports expressed in both values and volumes and global services exports, and over five different data vintages.

In addition to better empirical performance for the three target series, LSTMs provide advantages over DFMs by being able to handle large numbers of features without computational bottlenecks, not relying on the invertibility of any matrices, thus being able to be fit on any dataset, and the ability to use any mixture of frequencies in features or target. Disadvantages relative to DFMs include LSTMs’ stochastic nature and opacity regarding feature contribution to predictions.

The nowcast_lstm library can facilitate the use of LSTMs in economic nowcasting by lowering the barrier to experimentation. LSTMs’ ability to reliably generate predictions on a large number of input features makes it easier to quickly verify whether or not a given series has the potential to be nowcast, a characteristic that could help expand the variety and quantity of economic variables monitored via nowcasting.

There remains much scope for future research and development on this topic. Further testing should be performed to verify LSTMs’ performance on a wider variety of series and frequency mixtures. More hyperparameter tuning could be performed to see if tweaking other aspects of model architecture could result in better results. There is also much scope for exploring different methods of filling missing values beyond ARMA or mean-filling. Finally, methods in addition to the one already implemented for interpreting LSTMs and ascertaining feature contribution to predictions would increase the method’s viability as a policy-informing instrument. The library could be extended in the future to incorporate any improvements to performance or functionality deriving from future research, continuing to facilitate the adoption and development of the methodology in the nowcasting domain.

6. References

Amidi, A., and S. Amidi. 2019. “Recurrent Neural Networks Cheatsheet”. Recurrent Neural Networks Cheatsheet. 2019. Available at: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks. (accessed December 2020).
Antolin-Diaz, J., T. Drechsel, and I. Petrella. 2020. Advances in Nowcasting Economic Activity: Secular Trends, Large Shocks and New Data. DOI: http://dx.doi.org/10.2139/ssrn.3669854.
Banbura, M., D. Giannone, and L. Reichlin. 2010. Nowcasting. ECB Working Paper 1275. DOI: http://dx.doi.org/10.2139/ssrn.1717887.
Bańbura, M, and G. Rüstler. 2011. “A Look into the Factor Model Black Box: Publication Lags and the Role of Hard and Soft Data in Forecasting GDP”. *International Journal of Forecasting* 27 (2): 333–346. DOI: [https://doi.org/10.1016/j.ijforecast.2010.01.011](https://doi.org/10.1016/j.ijforecast.2010.01.011).

Bok, B., D. Caratelli, D. Giannone, A.M. Sbordone, and A. Tambalotti. 2018. “Macroeconomic Nowcasting and Forecasting with Big Data”. *Annual Review of Economics* 10 (1): 615–643. DOI: [https://doi.org/10.1146/annurev-economics-080217-053214](https://doi.org/10.1146/annurev-economics-080217-053214).

Brownlee, J. 2018. *Deep Learning for Time Series Forecasting: Predict the Future with MLPs, CNNs and LSTMs in Python*. Machine Learning Mastery. Available at: [https://books.google.ch/books?id=o5qnDwAAQBAJ](https://books.google.ch/books?id=o5qnDwAAQBAJ).

Buono, D., G. Mazzi, M. Marcellino, and Kapetanios. 2017. “Big Data Types for Macroeconomic Nowcasting”. *Eurona*.

Camacho, M., Y. Lovcha, and G. Perez Quiros. 2015. “Can We Use Seasonally Adjusted Variables in Dynamic Factor Models?” *Studies in Nonlinear Dynamics & Econometrics* 19 (3): 377–391. DOI: [https://doi.org/10.1515/snde-2013-0096](https://doi.org/10.1515/snde-2013-0096).

Cantú, F. 2018. *Estimation of a Coincident Indicator for International Trade and Global Economic Activity*. 27. UNCTAD Research Paper. UNCTAD. Available at: [https://unctad.org/system/files/official-document/ser-rp-2018d9_en.pdf](https://unctad.org/system/files/official-document/ser-rp-2018d9_en.pdf).

Chernis, T., and R. Sekkel. 2017. “A Dynamic Factor Model for Nowcasting Canadian GDP Growth”. *Empirical Economics* 53(1): 217–234. DOI: [https://doi.org/10.1007/s00181-017-1254-1](https://doi.org/10.1007/s00181-017-1254-1).

Chung, J., Ç. Gülçehre, K. Cho, and Y. Bengio. 2014. “Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling”. *CoRR* abs/1412.3555. Available at: [http://arxiv.org/abs/1412.3555](http://arxiv.org/abs/1412.3555).

De Veaux, R.D., and L.H. Ungar. 1994. “Multicollinearity: A Tale of Two Nonparametric Regressions”. In *Selecting Models from Data*, edited by P. Cheeseman and R.W. Oldford, 393–402. New York, NY: Springer New York.

Dematos, G., M.S. Boyd, B. Kermanshahi, N. Kohzadi, and I. Kaastra. 1996. “Feedforward versus Recurrent Neural Networks for Forecasting Monthly Japanese Yen Exchange Rates”. *Financial Engineering and the Japanese Markets* 3(1): 59–75. DOI: [https://doi.org/10.1007/BF00868008](https://doi.org/10.1007/BF00868008).

Domo. 2017. “Data Never Sleeps 5.0”. *Data Never Sleeps 5.0*. 2017. Available: [https://www.domo.com/learn/data-never-sleeps-5](https://www.domo.com/learn/data-never-sleeps-5). (accessed September 2021).

Doz, C., D. Giannone, and L. Reichlin. 2011. “A Two-Step Estimator for Large Approximate Dynamic Factor Models Based on Kalman Filtering”. *Journal of Econometrics* 164(1): 188–205. DOI: [https://doi.org/10.1016/j.jeconom.2011.02.012](https://doi.org/10.1016/j.jeconom.2011.02.012).

Einav, L., and J. Levin. 2014. “The Data Revolution and Economic Analysis”. *Innovation Policy and the Economy* 14: 1–24. DOI: [https://doi.org/10.1086/674019](https://doi.org/10.1086/674019).

Engle, R.F. 1982. “Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation”. *Econometrica* 50(4): 987–1007. DOI: [https://doi.org/10.2307/1912773](https://doi.org/10.2307/1912773).

Ennett, C.M., M. Frize, and C.R. Walker. 2001. “Influence of Missing Values on Artificial Neural Network Performance.” *Studies in Health Technology and Informatics* 84 (1): 449–53.
Falat, L., and L. Pancikova. 2015. “Quantitative Modelling in Economics with Advanced Artificial Neural Networks”. *Procedia Economics and Finance* 34: 194–201. DOI: https://doi.org/10.1016/S2212-5671(15)01619-6.

Fan, F., J. Xiong, and G. Wang. 2020. “On Interpretability of Artificial Neural Networks”. *CoRR* abs/2001.02522. Available at: http://arxiv.org/abs/2001.02522.

Federal Reserve Bank of New York. 2021. “Nowcasting Report: Methodology”. *Nowcasting Report: Methodology*. 2021. Available at: https://www.newyorkfed.org/research/policy/nowcast/methodology.html. (accessed November 2020)

Giannone, D., L. Reichlin, and S. Simonelli. 2009. “Nowcasting Euro Area Economic Activity in Real Time: The Role of Confidence Indicators”. *National Institute Economic Review* 210 (1): 90–97. DOI: https://doi.org/10.1177/0027950109354413.

Giannone, D., L. Reichlin, and D. Small. 2005. *Nowcasting GDP and Inflation: The Real Time Informational Content of Macroeconomic Data Releases*. Centre for Economic Policy Research. Available at: https://cepr.org/active/publications/discussion_papers/dp.php?dpno=5178.

Giannone, D., L. Reichlin, and D. Small. 2008. “Nowcasting: The Real-Time Informational Content of Macroeconomic Data”. *Journal of Monetary Economics* 55(4): 665–666. DOI: https://doi.org/10.1016/j.jmoneco.2008.05.010.

Grosse, R. 2017. *Lecture 15: Exploding and Vanishing Gradients*. Available at: http://www.cs.toronto.edu/~rgrosse/courses/csc321_2017/readings/L15%20Exploding%20and%20Vanishing%20Gradients.pdf. (accessed December 2020)

Guichard, S., and E. Rusticelli. 2011. *A Dynamic Factor Model for World Trade Growth*: 874. OECD Economics Department Working Papers. DOI: https://doi.org/10.1787/5kg9zbvwwq2-en.

Gurney, K. 1997. *An Introduction to Neural Networks*. USA: Taylor & Francis, Inc.

Hochreiter, S., and J. Schmidhuber. 1997. “Long Short-Term Memory”. *Neural Computation* 9: 1735–80. DOI: https://doi.org/10.1162/neco.1997.9.8.1735.

Hodas, N.O., and P. Stinis. 2018. “Doing the Impossible: Why Neural Networks Can Be Trained at All”. *Frontiers in Psychology* 9: 1185. DOI: https://doi.org/10.3389/fpsyg.2018.01185.

Hopp, D. 2021a. “nowcast_lstm”. Available at: https://github.com/dhopp1/nowcast_lstm

Hopp, D. 2021b. “nowcastLSTM”. Available at: https://github.com/dhopp1/nowcast_LSTM/

Hopp, D. 2021c. “nowcast_lstm_matlab”. Available at: https://github.com/dhopp1/nowcast_lstm_matlab/

Hopp, D. 2021d. “NowcastLSTM.jl”. Available at: https://github.com/dhopp1/NowcastLSTM.jl/

Johansen, S. 1995. *Likelihood-Based Inference in Cointegrated Vector Autoregressive Models*. New York.

Keskar, N.S., D. Mudigere, J. Nocedal, M. Smelyanskiy, and P.T.P. Tang. 2017. *On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima*. Available at: https://arxiv.org/abs/1609.04836.

Kozlov, M., S. Karaivanov, D. Tsonev, and R. Valkov. 2018. “The News on Norcasting”. *The News on Nowcasting*. Available at: https://www.weareworldquant.com/en/thought-leadership/the-news-on-nowcasting/. (accessed December 2020)
Kurihara, Y., and A. Fukushima. 2019. “AR Model or Machine Learning for Forecasting GDP and Consumer Price for G7 Countries”. Applied Economics and Finance 6: 1. DOI: https://doi.org/10.11114/aef.v6i3.4126.

Kuzin, V.N., M. Marcellino, and C. Schumacher. 2009. MIDAS versus Mixed-Frequency VAR: Nowcasting GDP in the Euro Area. Discussion Paper Series 1: Economic Studies 2009,07. Deutsche Bundesbank. Available at: https://ideas.repec.org/p/zbw/bubdp1/7576.html.

Loermann, J., and B. Maas. 2019. Nowcasting US GDP with Artificial Neural Networks. MPRA Paper 95459. University Library of Munich, Germany. Available at: https://ideas.repec.org/p/pra/mprapa/95459.html.

MacFeely, S. 2020. “In Search of the Data Revolution: Has the Official Statistics Paradigm Shifted?” Statistical Journal of the IAOS 36(4): 1075–1094. DOI: https://doi.org/10.3233/SJI-200662.

Marcellino, M., and C. Schumacher. 2010. “Factor MIDAS for Nowcasting and Forecasting with Ragged-Edge Data: A Model Comparison for German GDP”. Oxford Bulletin of Economics and Statistics 72(4): 518–550.

Mariano, R.S., and Y. Murasawa. 2003. “A New Coincident Index of Business Cycles Based on Monthly and Quarterly Series”. Journal of Applied Econometrics 18(4): 427–443. DOI: https://doi.org/10.1002/jae.695.

Matheson, T. 2011. “New Indicators for Tracking Growth in Real Time”. IMF Working Paper 11 (43): 1–22.

Molnar, C. 2019. Interpretable Machine Learning: A Guide for Making Black Box Models Explainable.

Montavon, G., W. Samek, and K.-R. Müller. 2018. “Methods for Interpreting and Understanding Deep Neural Networks”. Digital Signal Processing 73: 1–15. DOI: https://doi.org/10.1016/j.dsp.2017.10.011.

Morgado, A.J., L. Catela Nunes, and S. Salvador. 2007. Nowcasting an Economic Aggregate with Disaggregate Dynamic Factors: An Application to Portuguese GDP. GEE Papers 0002. Gabinete de Estratégia e Estudos, Ministério da Economia. Available at: https://ideas.repec.org/p/mde/wpaper/0002.html.

Nielsen, M.A. 2015. “Neural Networks and Deep Learning.” Determination Press.

Olah, C. 2015. Understanding LSTM Networks. Available at: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

OpenTable. 2021. “The State of the Restaurant Industry”. The State of the Restaurant Industry. Available at: https://www.opentable.com/state-of-industry. (accessed September 2021).

Porshakov, A., A. Ponomarenko, and A. Sinyakov. 2016. “Nowcasting and Short-Term Forecasting of Russian GDP with a Dynamic Factor Model”. Journal of the New Economic Association 30(2): 60–76.

PyTorch. 2021a. “Dropout”. Available at: https://pytorch.org/docs/stable/generated/torch.nn.Dropout.html. (accessed October 2021)

PyTorch. 2021b. “Loss Functions”. Available at: https://pytorch.org/docs/stable/n- n.html#loss-functions. (accessed August 2021).

PyTorch. 2021c. “LSTM”. Available at: https://pytorch.org/docs/stable/generated/torch.nn.LSTM.html. (accessed October 2021).
Rossiter, J. 2010. *Nowcasting the Global Economy*. 2010–2012. Bank of Canada. Available at: https://ssrn.com/abstract=1674952. DOI: http://dx.doi.org/10.2139/ssrn.1674952.

Rumelhart, D.E., G.E. Hinton, and R.J. Williams. 1986. “Learning Representations by Back-Propagating Errors”. *Nature* 323 (6088): 533–36. DOI: https://doi.org/10.1038/323533a0.

Scikit-learn. 2021. “3.1. Cross-Validation: Evaluating Estimator Performance”. 3.1. Cross-Validation: Evaluating Estimator Performance. 2021. Available at: https://Scikit-learn.org/stable/modules/cross_validation.html. (accessed October 2021)

Sharma, S., S. Sharma, and A. Athaiya. 2020. “Activation Functions in Neural Networks”. *International Journal of Engineering Applied Sciences and Technology*, 4(12): 310–316.

Shi, X., Z. Chen, H. Wang, D.-Y. Yeung, W. Wong, and W. Woo. 2015. *Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting*. Smieja, M.L. Struski, J. Tabor, B. Zieliński, and P. Spurek. 2019. *Processing of Missing Data by Neural Networks*.

Statista. 2021. “Coronavirus Impact on Retail E-Commerce Website Traffic Worldwide as of June 2020, by Average Monthly Visits”. Coronavirus Impact on Retail E-Commerce Website Traffic Worldwide as of June 2020, by Average Monthly Visits. 2021. Available at: https://www.statista.com/statistics/1112595/covid-19-impact-retail-e-commerce-site-traffic-global/. (accessed March 2021).

Stock, J.H., and M.W. Watson. 2002. “Forecasting Using Principal Components From a Large Number of Predictors”. *Journal of the American Statistical Association* 97(460): 1167–1179. DOI: https://doi.org/10.1198/016214502388618960.

Stock, J.H., and M.W. Watson. 2004. “Combination Forecasts of Output Growth in a Seven-Country Data Set”. *Journal of Forecasting* 23(6): 405–430. DOI: https://doi.org/10.1002/for.928.

Stratos, K. 2020. *Feedforward and Recurrent Neural Networks*. Available at: http://www1.cs.columbia.edu/~stratos/research/neural.pdf. (accessed December 2020)

Stratos, K. 2020. “Transforming Our World: The 2030 Agenda for Sustainable Development”. *Transforming Our World: The 2030 Agenda for Sustainable Development*. Available at: https://sdgs.un.org/2030agenda.

UN. 2015. “Transforming Our World: The 2030 Agenda for Sustainable Development”. *Transforming Our World: The 2030 Agenda for Sustainable Development*. Available at: https://sdgs.un.org/2030agenda. (accessed September 2020).

UNCTAD. 2020a. “UNCTADStat”. UNCTADStat. 2020. Available at: https://unctadstat.unctad.org/EN/Index.html. (accessed March 2021).

UNCTAD. 2020b. “Global Merchandise Trade Nowcast December 2020”. *Global Merchandise Trade Nowcast December 2020*. Available at: https://unctad.org/system/files/official-document/gdssdsimisc2020d8_en.pdf. (accessed August 2020).

UNSD. 2020. “Nowcasting and Forecasting for SDG Monitoring”. Presented at the Nowcasting and Forecasting for SDG Monitoring, February 3, Geneva, Switzerland. Available at: https://unstats.un.org/unsd/statcom/51st-session/side-events/20200302-2L-Nowcasting-and-Forecasting-for-SDG-Monitoring/. (accessed February 2021).
“The X-13ARIMA-SEATS Seasonal Adjustment Program”. The X-13ARIMA-SEATS Seasonal Adjustment Program. 2017. Available at: https://www.census.gov/srd/www/x13as/.

USCB. 2017. “The X-13ARIMA-SEATS Seasonal Adjustment Program”. The X-13ARIMA-SEATS Seasonal Adjustment Program. 2017. Available at: https://www.census.gov/srd/www/x13as/. (accessed March 2021).

WMO. 2017. “Guidelines for Nowcasting Techniques”. 1198. WMO. Available at: https://library.wmo.int/doc_num.php?explnum_id = 3795. (accessed March 2021).

WTO. 2020. “Statistics on Merchandise Trade”. Statistics on Merchandise Trade. 2020. Available at: https://www.wto.org/english/res_e/statis_e/merch_trade_stat_e.htm. (accessed November 2020).

Yilmazkuday, Hakan. 2021. “Stay-at-Home Works to Fight against COVID-19: International Evidence from Google Mobility Data”. Journal of Human Behavior in the Social Environment 31 (1–4): 210–220. DOI: https://doi.org/10.1080/10911359.2020.1845903.

Received July 2021
Revised August 2021
Accepted February 2022