Journal of Econometrics xxx (xxxx) xxx

Contents lists available at ScienceDirect

Journal of Econometrics

journal homepage: www.elsevier.com/locate/jeconom

Bayesian neural networks for macroeconomic analysis[☆]

Niko Hauzenberger[a], Florian Huber[b], Karin Klieber[c], Massimiliano Marcellino[d,*]

[a] University of Strathclyde, United Kingdom
[b] University of Salzburg, Austria
[c] Oesterreichische Nationalbank, Austria
[d] Bocconi University, IGIER, CEPR, Baffi-Carefin and BIDSA, Italy

ARTICLE INFO

JEL classification:
C11
C30
C45
C53
E3
E44

Keywords:
Bayesian neural networks
Model selection
Shrinkage priors
Macroeconomic forecasting

ABSTRACT

Macroeconomic data is characterized by a limited number of observations (small T), many time series (big K) but also by featuring temporal dependence. Neural networks, by contrast, are designed for datasets with millions of observations and covariates. In this paper, we develop Bayesian neural networks (BNNs) that are well-suited for handling datasets commonly used for macroeconomic analysis in policy institutions. Our approach avoids extensive specification searches through a novel mixture specification for the activation function that appropriately selects the form of nonlinearities. Shrinkage priors are used to prune the network and force irrelevant neurons to zero. To cope with heteroskedasticity, the BNN is augmented with a stochastic volatility model for the error term. We illustrate how the model can be used in a policy institution through simulations and by showing that BNNs produce more accurate point and density forecasts compared to other machine learning methods.

1. Introduction

Policy making in central banks and other governmental institutions is often informed by economic models that are, for simplicity, assumed to be linear or take relatively simple nonlinear forms. For instance, the Phillips curve is a key analytical framework for the analysis and conduct of monetary policy. This relationship is often assumed to be linear and empirical specifications have been shown to forecast poorly (see, e.g., Clark and McCracken, 2006). The weak out-of-sample predictive power is often attributed to structural breaks in the coefficients or other forms of nonlinearities and researchers increasingly adopt nonlinear parametric models to estimate Phillips curves (see, e.g., Benigno and Eggertsson, 2023).

The main obstacle to using such flexible models, particularly in policy institutions, is that they often require strong assumptions on the nature of nonlinearities and this calls for substantial prior information. The question, however, is whether a particular form of nonlinearity is really appropriate or whether it implies model mis-specification. Deciding on the appropriate form of nonlinearities is thus crucial for accurate inference, and modern machine learning techniques can be applied to learn the functional form from the data.

☆ The authors thank the Editors Serena Ng and George Kapetanios, three anonymous referees, Jamie Cross, Martin Feldkircher, Laurent Ferrara, Christian Hotz-Behofsits, Malte Knueppel, Michael McCracken, Carlos Montes-Galdon, Michael Pfarrhofer, and participants at the CFE 2022 conference, COMPSTAT 2022 conference, 12th ECB Conference on Forecasting Techniques 2023, and IIF MacroFor 2023 webinar for helpful comments and suggestions. Hauzenberger gratefully acknowledges financial support from the Jubiläumsfonds of the Oesterreichische Nationalbank (OeNB, grant no. 18718, 18763, and 18765), and Huber acknowledges financial support from the Austrian Science Fund (FWF, grant no. ZK 35) and the Jubiläumsfonds of the OeNB (grant no. 18304). The views expressed in this paper do not necessarily reflect those of the OeNB or the Eurosystem.

* Correspondence to: Department of Economics, Bocconi University, Via Roentgen 1, 20136 Milano, Italy.
E-mail address: massimiliano.marcellino@unibocconi.it (M. Marcellino).

https://doi.org/10.1016/j.jeconom.2024.105843
Received 5 April 2023; Received in revised form 18 August 2024; Accepted 19 August 2024

In this paper, we develop Bayesian neural networks (BNNs) for macroeconomic policy analysis. NNs have the advantage of being able to approximate any form of nonlinear conditional mean relation arbitrarily well (Hornik et al., 1989) and — in a wide range of different fields (see, e.g., Kourentzes, 2013; Wen et al., 2017; Salinas et al., 2020; Sezer et al., 2020) — have been shown to be particularly useful for forecasting purposes. Moreover, they nest other popular methods in machine learning such as tree models (see Wang and Rockova, 2020). Their use in econometrics has been limited (some recent relevant examples include Farrell et al., 2021; Chronopoulos et al., 2024, 2023). This is due to several reasons, and we aim to address (at least) two of these through our model.

First, NNs are trained on vast datasets and often feature millions of free parameters. For instance, the MNIST dataset (LeCun et al., 1998) includes $T = 60{,}000$ observations and $K = 784$ covariates. By contrast, macroeconomic datasets such as the popular McCracken and Ng (2016) database feature a few hundred observations and almost as many possible time series. Hence, specifying and estimating fully-fledged multi-layer NNs on such datasets is challenging, which makes their use for empirical analyses and policy making problematic. For instance, applications require to specify the number of hidden layers, the number of neurons, the form of the activation function, the algorithm, as well as associated nuisance parameters used for training the models.

To address these issues, we aim to minimize the possible range of competing choices by introducing stochastic model selection techniques to decide both the type of (layer-specific) activation functions and the number of neurons per layer. The former is achieved through a novel mixture model that averages over a set of pre-specified activation functions. The vast majority of NN papers assume instead a single activation function per layer, and a common choice with good theoretical properties is the Rectified Linear Unit (ReLU, see Polson and Ročková, 2018; Farrell et al., 2021). Yet, this choice might be restrictive, and our mixture specification provides additional flexibility, without the need to carry out cross-validation or repeated estimation of the model. Moreover, we select the number of neurons by using Bayesian shrinkage priors that force individual neurons aggressively to zero. In particular, we follow Ghosh et al. (2019) and Bhadra et al. (2020) and use a horseshoe prior which does not depend on additional tuning parameters. As a result, our proposed BNN only requires the researcher to decide on the number of hidden layers.

Second, to speed up computations, NNs are typically estimated using maximum a-posteriori (MAP), minimum mean squared error (MMSE), or variational Bayes (VB) inference. However, uncertainty quantification with these techniques is difficult and often relies on approximations. For instance, VB only approximates the joint posterior distribution of the model and neglects any parameter uncertainty in the predictive distributions, resulting in underestimating the actual predictive variance. Moreover, capturing departures from non-Gaussianity (such as heavy tails, multi-modality, or skewness) would also be challenging. To cope with this, we opt for a fully Bayesian estimation approach that samples from the exact full conditional posterior distributions.[1] In our model, objects of interest, such as predictive distributions, can then be obtained through Monte Carlo integration. The resulting predictive densities can be highly non-Gaussian and exhibit features such as heavy tails, multiple modes, or skewness.

After developing theoretical tools for the specification, estimation and forecasting for BNNs, we assess their empirical performance, first with synthetic data, and then with actual US macro data. To generate synthetic data that closely resemble actual macroeconomic dynamics, we use a data generating process (DGP) inspired by the nonlinear Phillips curve in Benigno and Eggertsson (2023). It turns out that shallow models with flexible activation functions recover the mean function of inflation with more precision than deep models.

Next, we show that our BNNs produce accurate short-term forecasts for key US economic variables (inflation, industrial production, and employment). These are often better than those from standard models commonly used in econometrics (such as dynamic models augmented with principal components and unobserved component models) and machine learning (such as the LASSO, elastic net, (Bayesian) additive regression trees, and random forests). In addition, for all the variables we consider, deep BNNs produce a much better in-sample fit than shallow BNNs, but differences in density forecast accuracy are very small, which supports the use of the latter which are computationally simpler. Moreover, our mixture activation function yields generally only modest gains, but it frees the researcher from the necessity to decide on one particular activation function and thus reduces the number of inputs to the model. Furthermore, a recursive evaluation suggests that nonlinearities pay off during turbulent times, such as the recession in the early 2000s, the global financial crisis (GFC), and the onset of the Covid-19 pandemic.

The remainder of the paper is structured as follows. Section 2 develops our BNN model for use with macroeconomic data. Section 3 illustrates the model using synthetic data that resemble US inflation dynamics. Section 4 considers forecasting US economic variables. The last section summarizes and concludes the paper. An Online Appendix provides additional technical details and empirical findings.

2. Bayesian neural networks for macroeconomic data

This section develops our general BNN model. After discussing key model specification issues in Section 2.1, we consider approximations in Section 2.2, choice of the activation function in Section 2.3, modeling the error variance in Section 2.4, suitable Bayesian regularization priors in Section 2.5, and posterior computation in Section 2.6.

[1] Exact conditional on the Markov chain Monte Carlo (MCMC) algorithm to have converged to the correct stationary distribution.

2.1. A general nonparametric regression model

Our goal is to estimate an unknown and possibly nonlinear relationship between a macroeconomic time series $y_t \in \mathbb{R}$ and K covariates $\boldsymbol{x}_t \in \mathbb{R}^K$. In what follows, the covariates in $\boldsymbol{x}_t$ can include lags of y_t as well as other (lagged) macroeconomic and financial indicators. In our setting, K can be very large relative to the sample size T and the relationship between y_t and $\boldsymbol{x}_t$ possibly highly nonlinear.

We assume a general nonlinear regression given by:

$$y_t = \boldsymbol{x}_t'\boldsymbol{\gamma} + f(\boldsymbol{x}_t) + \varepsilon_t, \quad \varepsilon_t \sim \mathcal{N}(0, \sigma_t^2), \tag{1}$$

where $\boldsymbol{\gamma}$ is a vector of K linear coefficients, $f : \mathbb{R}^K \to \mathbb{R}$ is a function of unknown (nonlinear) form, and ε_t is a Gaussian shock with zero mean and time-varying variance σ_t^2. A typical assumption is that f is α-Hölder smooth on $[0,1]^K$.[2] The class of α-Hölder smooth functions is defined by $\mathcal{H}_K^\alpha = \{f : [0,1]^K \to \mathbb{R}; \|f\|_{\mathcal{H}}^\alpha < \infty\}$, where $\|f\|_{\mathcal{H}}^\alpha$ denotes the Hölder norm (Polson and Ročková, 2018; Schmidt-Hieber, 2020). The parameter $\alpha > 0$ indicates the smoothness of the class of functions we aim to approximate.

The inclusion of a linear part in the model is meant to use the nonlinear component only to capture proper nonlinear relationships between the target and the explanatory variables. Recent evidence suggests that linear models are competitive in terms of forecast accuracy. It is mostly during turbulent times (such as recessions) that adding nonlinear components pays off (see, e.g., Clark et al., 2023). We take this empirical fact into account through the linear piece which we expect to explain the vast majority of variation over the estimation period.

Similarly, the presence of a time-varying error variance reduces the risks that nonlinearities in the conditional mean show up simply to capture outliers or periods of high volatility. It also implies that our model adapts to situations not learned through the conditional mean by increasing σ_t^2, thus inflating uncertainty surrounding predictions to provide a proper assessment of the point forecasts reliability, which matters particularly when the latter are used for policy making. The assumption of Gaussian shocks is not restrictive when combined with stochastic volatility (SV), but it would be feasible to incorporate more flexible error distributions based on scale-location mixtures of Gaussians (see, e.g., Escobar and West, 1995).

In macroeconomics, f is often assumed to be known. For instance, if $f(\boldsymbol{x}_t) = 0$ for all t we end up with a constant parameter regression model. Another commonly used model arises if $f(\boldsymbol{x}_t) = \boldsymbol{x}_t'\boldsymbol{\gamma}_t$ with $\boldsymbol{\gamma}_t$ denoting K time-varying parameters (TVPs). Other specifications which can be seen as special cases of Eq. (1) are threshold and Markov switching models (see, e.g., Hamilton, 1989; Tong, 1990; Teräsvirta, 1994), polynomial regression (see, e.g., McCrary, 2008; Lee and Lemieux, 2010) or models with interaction effects (see, e.g., Ai and Norton, 2003; Imbens and Wooldridge, 2009; Greene, 2010).

This brief discussion shows that the choice of f is one of the most important modeling decisions the researcher needs to take. In this paper, we follow a different route and estimate f. This can be achieved through nonparametric techniques such as Bayesian additive regression trees (see, e.g., Chipman et al., 2010; Huber et al., 2023; Clark et al., 2023, 2024), random forests (see, e.g., Goulet Coulombe, 2024), Gaussian processes (see, e.g., Williams and Rasmussen, 2006; Crawford et al., 2019; Hauzenberger et al., 2024), splines (see, e.g., Vasicek and Fong, 1982; Engle and Rangel, 2008), or wavelets (see, e.g., Ramsey and Lampart, 1998; Gallegati, 2008).

2.2. Deep neural network approximation

In this paper, we aim to learn f using a NN. In many areas outside economics, NNs have been used with some success (see Lara-Benítez et al., 2021; Lim and Zohren, 2021, for surveys on time series forecasting with deep learning). The goal of this paper is to design NNs that can be used in combination with macroeconomic data and its associated particularities. These relate to the fact that the data is often persistent, features structural breaks and volatility clustering but also that the length of the time series is short. Hence, researchers and practitioners alike cannot rely on theoretical results in the literature (see, e.g., Polson and Ročková, 2018; Schmidt-Hieber, 2020; Farrell et al., 2021) that mostly assume T to be huge but also rule out autocorrelation in the data. Hence, the large-scale NNs that are typically used are often not suited for macro datasets, and this motivates the main modeling choices we take.

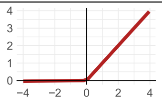
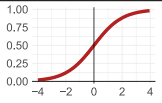
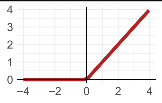
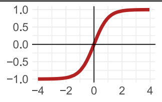
We use a deep NN to approximate the function f. This approximating model consists of L layers, with each layer featuring Q_ℓ $(\ell = 1, \ldots, L)$ neurons. These neurons recursively produce outputs by taking nonlinear transformations of inputs (which, at the first layer, are simply the covariates in $\boldsymbol{x}_t$). The nonlinear transformations are defined by an activation function $h_{\ell,q}$ $(q = 1, \ldots, Q_\ell)$. In what follows, we let $\boldsymbol{h}_\ell = (h_{\ell,1}, \ldots, h_{\ell,Q_\ell})'$ denote an activation function that can be layer and neuron-specific.[3] In this case, our approximation to f can be written as:

$$f(\boldsymbol{x}_t) \approx \hat{f}_L(\boldsymbol{x}_t) = \boldsymbol{W}_{L+1}\boldsymbol{h}_L\left(\boldsymbol{W}_L\boldsymbol{h}_{L-1}(\cdots \boldsymbol{W}_2\boldsymbol{h}_1(\boldsymbol{W}_1\boldsymbol{x}_t))\right), \tag{2}$$

[2] The assumption that $\boldsymbol{x}_t$ is restricted to the K-dimensional unit hypercube is not restrictive in practice since we can appropriately transform any covariate to fulfill this.

[3] In the discussion we do not add bias terms to simplify notation. In a neural network, the bias term can be defined as the constant that allows the activation function to be shifted towards positive and negative values. In our empirical work, all networks include a bias term as well.

Table 1
Set of activation functions.

	Activation function	Equation	Plot
(1)	LeakyReLU	$h^{(1)}(x) = \begin{cases} 0.01x & x < 0 \\ x & x \geq 0 \end{cases}$	
(2)	Sigmoid	$h^{(2)}(x) = \frac{1}{1+\exp(-x)}$	
(3)	Rectified Linear Unit (ReLU)	$h^{(3)}(x) = \max(0, x)$	
(4)	Hyperbolic tangent (tanh)	$h^{(4)}(x) = \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}$	

with $\boldsymbol{W}_\ell$ denoting a $Q_\ell \times Q_{\ell-1}$ matrix of network weights for $\ell = 2, \dots, L$, while $\boldsymbol{W}_{L+1}$ is a $1 \times Q_L$ vector and $\boldsymbol{W}_1$ is a $Q_1 \times K$ matrix. In the subsequent discussion, we set $Q_\ell = Q$ for $\ell = 1, \dots, L$. It is worth noting that if $L = 1$, we end up with a shallow neural network:

$$\hat{f}_1(\boldsymbol{x}_t) = \boldsymbol{W}_2 \boldsymbol{h}_1(\boldsymbol{W}_1 \boldsymbol{x}_t).$$

Both shallow and deep neural networks serve as universal approximators. Deep NNs, as outlined in Mhaskar et al. (2017), can achieve a certain level of approximation error with exponentially fewer parameters than shallow NNs. If the true function f is compositional, meaning that $f(\boldsymbol{x}_t) = h_J(h_{J-1}(\dots h_1(\boldsymbol{x}_t)))$, a shallow neural network would require a very large number of neurons to approximate it with an arbitrary approximation error $c > 0$. For the macro series we are interested in, it is not clear whether the true underlying structural model implies a compositional structure and, in addition, given that the informational content in typical datasets is relatively low compared to the vast dimensional datasets used in other fields, we will treat the shallow NN as a serious competitor in the analysis that follows.

Typically, researchers treat the weights as the only unknown parameters in the network and set L and $\boldsymbol{h}_l$ a priori and based on best practice or cross-validation. The choice of the activation function could have important consequences for the empirical performance of the model (Karlik and Olgac, 2011; Agostinelli et al., 2014) and, in our general specification, the activation function is not only layer- but also neuron-specific. Hence, the space of possible combinations of activation functions for a moderate number of hidden layers becomes intractable. A practicable solution is to set $h_{\ell,q} = h$ for all ℓ and q. In the literature, a standard choice with excellent theoretical properties is the Rectified Linear Unit activation function $\text{ReLU}(x) = \max(0, x)$. However, as stated above, theoretical results in, e.g., Polson and Ročková (2018) or Farrell et al. (2021) are based on situations where $T \to \infty$ and without assuming dependence in the responses. For macroeconomic data that are often analyzed in actual policy contexts, these conditions are not fulfilled and the small sample properties of the ReLU activation functions are not well understood. As a solution, one contribution of this paper is to treat $h_{\ell,q}$ as a discrete hyperparameter which we estimate alongside the weights $\boldsymbol{W}_\ell$ and other model parameters. We turn to this issue in the next sub-section.

2.3. Convex combinations of activation functions

Instead of using a single activation function for the whole network, we construct a mixture specification that averages over four commonly used activation functions: leakyReLU (1), sigmoid (2), ReLU (3), and hyperbolic tangent (tanh, 4). We let $h^{(m)}$ with $m \in \{\text{leakyReLU}, \text{sigmoid}, \text{ReLU}, \text{tanh}\}$ denote an activation function out of this set of different functions. Each specific choice $h^{(m)}$ has different implications on the flexibility of the neural network to capture nonlinearities in the data. Table 1 provides a summary of the functions used.

We assume that each activation function is given by:

$$h_{\ell,q}(z_{\ell q,t}) = \sum_{m=1}^{4} \omega_{\ell,q}^{(m)} h^{(m)}(z_{\ell q,t}), \tag{3}$$

with $z_{\ell q,t}$ being the qth element in the recursively defined vector $\boldsymbol{z}_{\ell,t} = \boldsymbol{W}_\ell \boldsymbol{h}_{\ell-1}(\boldsymbol{z}_{\ell-1,t})$ and $\boldsymbol{z}_{1,t} = \boldsymbol{W}_1 \boldsymbol{x}_t$. The neuron and layer-specific weights $\omega_{\ell,q}^{(m)}$ satisfy $\omega_{\ell,q}^{(m)} \geq 0$ and $\sum_m \omega_{\ell,q}^{(m)} = 1$. The main implication of this specification is that we do not need to decide on one particular activation function but we average over a set of prominent activation functions. The weights reflect the relative importance of a specific activation function over other options. In principle, if a researcher has prior knowledge that a given activation function should be nonlinear (either through observing features of the data or theoretical knowledge), one can use this information through a suitable prior on the weights $\omega_{\ell,q}^{(m)}$.

An alternative way of writing this mixture specification is through a discrete random variable $\delta_{\ell,q} \in \{1, \ldots, 4\}$. The probability that $\delta_{\ell,q} = m$ is then set equal to:

$$\text{Prob}(\delta_{\ell,q} = m) = \omega_{\ell,q}^{(m)}.$$

Using mixtures of activation functions increases the flexibility substantially without introducing a large number of additional parameters. Our approach is related to evolutionary neural networks (Yao, 1999; Turner and Miller, 2014) that learn the network structure adaptively but, as we will discuss below, allows for fully-fledged uncertainty quantification on the activation functions and the weights.

Depending on the choice of the weights, we can get combined activation functions that share features of each of the four individual activation functions. We illustrate the effect different activation functions have on the function estimates using a simple univariate example. This example models the relationship between US year-on-year inflation (y_t) and the year-on-year money growth rate (x_t) in a nonlinear manner. These two series are obtained from the FRED-MD database (McCracken and Ng, 2016). To account for the leading effect of money growth on inflation, we specify x_t as the 18th lag of money growth (see, e.g., Reichlin and Lenza, 2007; Amisano and Fagan, 2013). The corresponding nonparametric regression is given by:

$$y_t = f(x_t) + \varepsilon_t, \quad \varepsilon_t \sim \mathcal{N}(0, \sigma^2). \tag{4}$$

We compare the effect that different activation functions have on the mean estimate $f(x_t)$ in Fig. 1 by considering two BNNs estimated using the techniques outlined in the next sub-section. The first is a shallow one that sets $L = 1$ and includes only a single neuron. The red shaded areas represent the 5th and 95th percentiles of the posterior of $\widehat{f}_1(x_t)$ while the red line is the posterior median. The second model is a deep BNN with $L = 3$ and a single neuron for all ℓ as well. The corresponding posterior percentiles are depicted in blue. In this figure, the (lagged) values of money growth are on the x-axis while the yearly change in inflation is on the y-axis. Considering the linear specification (i.e., $f(x_t) = \beta x_t$) suggests a positive relationship between money growth and inflation. Nonlinear activation functions, both for the deep and shallow BNN, often yield estimates of mean relations that display substantial nonlinearities. This finding holds for tanh, ReLU and leakyReLU. There are cases where the deep model produces more nonlinearities (such as for tanh) or where thresholds change (in the case of ReLU) but the shape is often similar and suggests that for low levels of money growth, inflation displays only modest reactions. Notice that for leakyReLU, both the deep and shallow BNN produce similar mean relations, indicated by overlapping posterior credible intervals.

The differences between shallow and deep models increase if we consider our convex mixture activation function. In this case, the shallow neural network produces a fit close to the linear model. Notice, however, that for large levels of money growth, the uncertainty surrounding the functional estimates increases appreciably. By contrast, the deep BNN with flexible activation functions produces an estimate of the conditional mean that implies increasing levels of inflation for money growth between three and eight percent. For values larger than eight percent, the relationship flattens out and the model predicts no substantial fluctuations of inflation.

This short, stylized example illustrates that the different activation functions give rise to different, albeit similar, estimated mean relationships. Since in all our empirical work we set Q to a large value and use a large panel of covariates, the models we propose are capable of extracting complex nonlinear features in a very flexible manner.

2.4. Adding stochastic volatility

Neural networks often explicitly or implicitly assume that the error variance is constant. This assumption implies that the mean function explains a constant share of variation in y_t over time. For macroeconomic data, this assumption is strong. In exceptional periods such as the GFC or the Covid-19 pandemic, not only mean relations can change but larger shocks than usual can hit the economy. Hence, we model the error variance in a time-varying manner using a standard SV model.

Our model assumes that $v_t = \log \sigma_t^2$ evolves according to an AR(1) process:

$$v_t = \mu_v + \rho_v(v_{t-1} - \mu_v) + \varsigma_t, \quad \varsigma_t \sim \mathcal{N}(0, \xi_v^2), \tag{5}$$

with μ_v denoting the long-run level of the log-volatility, ρ_v the persistence parameter, and ξ_v^2 the state equation variance. This model assumes that the error variance evolves rather smoothly over time and features its own shock. For later convenience, we let $\boldsymbol{v} = (v_1, \ldots, v_T)'$ denote the full history of the log-volatilities and $\boldsymbol{\beta}_v = (\mu_v, \rho_v, \xi_v^2)'$ the parameters of the log-volatility state equation.

It is worth stressing that we opt for a SV specification because it is flexible and has been shown to work well for macroeconomic data (Clark, 2011). Alternative approaches such as the Generalized Autoregressive Conditional Heteroskedastic model (GARCH, Bollerslev, 1986) are also feasible. In principle, it would also be possible to use a NN to estimate the volatility process as well (see, e.g., Goulet Coulombe et al., 2023; Nguyen et al., 2023).

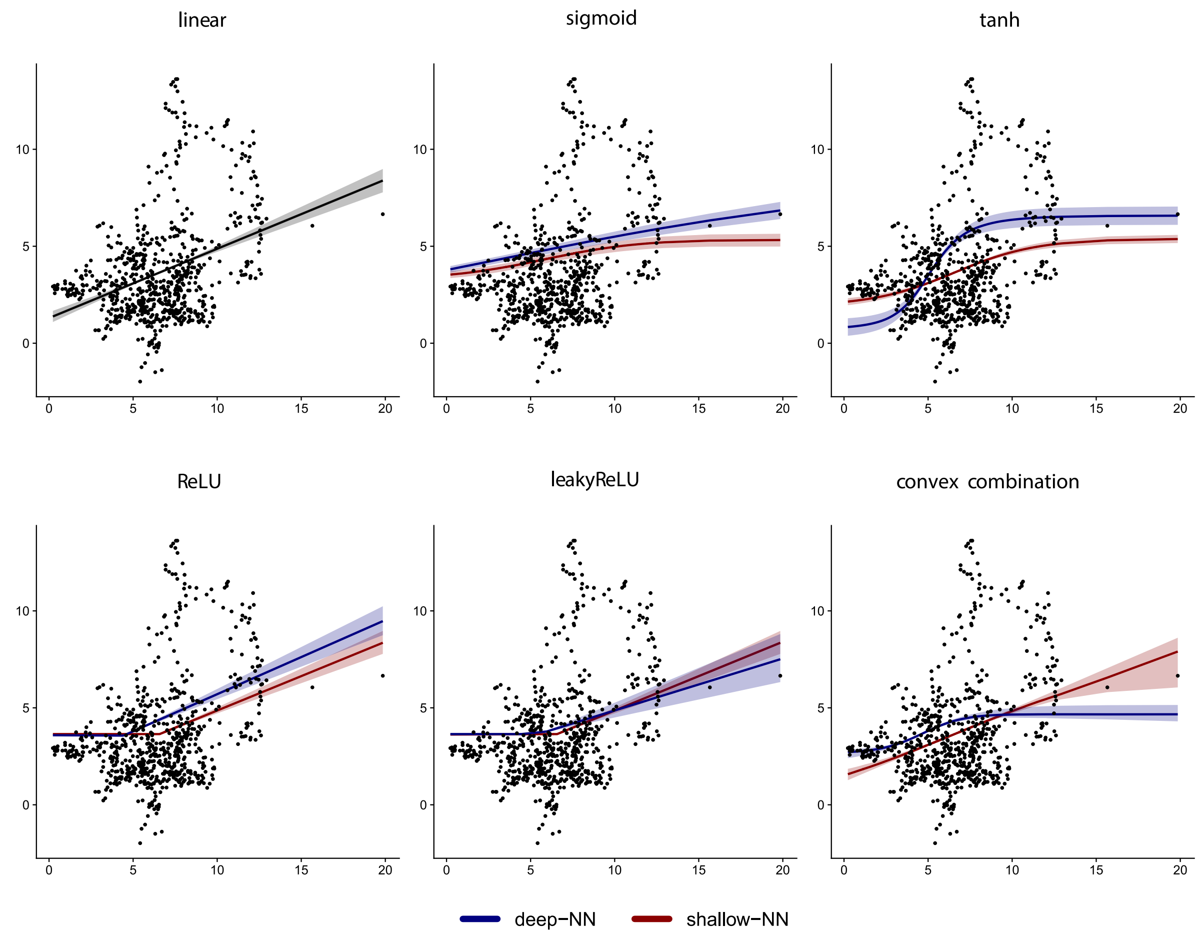


Fig. 1. Nonlinearities in the nexus between inflation and money growth. *Note:* This figure shows the nexus between inflation and money growth for the US and illustrates the functional forms of the activation functions specified in Table 1. The data for the consumer price index (i.e., CPIAUCSL) and money supply (i.e., M2SL) are taken from the FRED-MD database as described in McCracken and Ng (2016). We plot the following example: $y_t = f(x_t) + \epsilon_t$, where y_t is the year-on-year inflation and x_t is the 18th lag of the year-on-year money supply growth (see Eq. (4)). f refers to the activation functions specified in Table 1. The top left panel refers to the linear regression model while bottom-right panel 'convex combination' refers to our main specification, where we let the data decide on the form of the activation function. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

2.5. *Achieving shrinkage in deep neural networks*

We opt for a Bayesian approach to estimating the NN. This involves specifying suitable priors on the parameters of the model. The prior setup we use introduces regularization on the weights so that we can effectively use a large number of neurons per hidden layer but avoid overfitting issues by shrinking weights associated with irrelevant neurons to zero.

On the weights of the network we consider a shrinkage prior. Early contributions propose shrinkage and regularization priors on regression models that imply nonlinear transformations of the covariates but linearity in the parameters (see, e.g., Tipping, 2001). Titterington (2004) discusses shrinkage priors in NNs and comes up with a prior on the weights of the form:

$$p(\boldsymbol{W}_\ell|\phi_{\ell,ij}) \propto \exp\left\{-\frac{1}{2}\sum_i\sum_j \phi^2_{\ell,ij} w_{\ell,ij}\right\},$$

with $\phi_{\ell,ij}$ denoting a shrinkage hyperparameter that controls the degree of shrinkage. Hence, large values of $\phi_{\ell,ij}$ implies that the (i,j)th element of $\boldsymbol{W}_\ell, w_{\ell,ij}$, should be strongly forced to zero. Empirically, we often face a situation where most $\phi_{\ell,ij}$s are large and thus many elements in $\boldsymbol{W}_\ell$ are shrunk to zero, reducing the effective dimension of the state space considerably (Titterington, 2004).

To capture this, global local shrinkage (GL) priors (Polson and Scott, 2010) can be employed. These priors include a global shrinkage component that applies to groups of parameters and local components that allow for deviations in case of strong global shrinkage. A particular, hyperparameter-free, version of such a GL prior is the horseshoe prior (Carvalho et al., 2009). The horseshoe has been first applied to NNs in Ghosh et al. (2019) and Bhadra et al. (2020). This prior is used to flexibly prune NNs without requiring prior information or introducing additional hyperparameters.

Let $\boldsymbol{w}_{\ell,i\bullet}$ denote the ith row of $\boldsymbol{W}_\ell$ ($\ell = 1, \ldots, L{+}1$). Both on the linear coefficients in $\boldsymbol{\gamma}$ and $\boldsymbol{w}_{\ell,i\bullet}$ we use horseshoe priors (Carvalho et al., 2010). The horseshoe implies the following prior hierarchy on each element of $\boldsymbol{w}_{\ell,i\bullet} = (w_{\ell,i1}, \ldots, w_{\ell,iQ_{\ell-1}})'$:

$$w_{\ell,ij} \sim \mathcal{N}(0, \phi_{\ell,ij}), \quad \phi_{\ell,ij} = \lambda^2_{\ell,i}\varphi^2_{\ell,ij}, \quad \lambda_{\ell,i} \sim \mathcal{C}^+(0,1), \quad \varphi_{\ell,ij} \sim \mathcal{C}^+(0,1),$$

with $\lambda^2_{\ell,i}$ being a global (neuron-specific) shrinkage parameter which forces all elements in $\boldsymbol{w}_{\ell,i\bullet}$ towards the origin, $\varphi_{\ell,ij}$ is a local scaling parameter that allows for coefficient-specific deviations in light of strong global shrinkage (i.e., if $\lambda^2_{\ell,i} \approx 0$). Hence, within each layer we use the horseshoe to shrink the weights associated with irrelevant neurons to zero, effectively reducing the number of neurons per layer in a flexible manner.

It is worth stressing that there is a close relationship between using a shrinkage prior on the weights and dropout, another popular technique for NN regularization. Nalisnick et al. (2019) discuss the equivalence between shrinkage priors and dropout.

Next, we need to discuss the prior on the activation weights $\omega^{(m)}_{\ell,q}$. In this case, we specify the prior probability that $\text{Prob}(\delta_q = m) = 1/4$. This choice implies that each activation function is equally likely a priori. In principle, alternative choices would be possible so that more weight is put on functions such as the ReLU where there exists strong evidence that this choice works well empirically.

Finally, the prior on the parameters driving the SV processes are set along the lines suggested in Kastner and Frühwirth-Schnatter (2014). This amounts to eliciting a Gaussian prior on $\mu_v \sim \mathcal{N}(0, 10)$, a Beta prior on $(\rho_v + 1)/2 \sim \mathcal{B}(25, 1.5)$, and a Gamma prior on $\xi^2_v \sim \mathcal{G}(1/2, 1/(2s_v))$ where $s_v = 0.01$. The prior on the unconditional mean is relatively uninformative whereas the prior on the (transformed) persistence parameter pushes the latent process towards a random walk. The Gamma prior on ξ^2_v is equivalent to a Normal prior on $\pm\xi_v$ with variance s_v. The choice $s_v = 0.01$ implies some shrinkage on the amount of heteroskedasticity in the shocks.

2.6. *Posterior simulation*

In general, posterior inference in BNNs is extremely challenging. These challenges arise from the fact that typical input datasets are huge dimensional in T and K. In our case, given the limited length of the time series, estimating a BNN is possible and estimation of such a model can be done within an hour on a standard desktop computer.

At a general level, we draw from the joint posterior distribution using a Markov chain Monte Carlo (MCMC) algorithm that consists of several blocks. The most challenging block is related to the network weights for layers $\ell = 1, \ldots, L$. In this case, we use state-of-the-art Hamiltonian Monte Carlo (HMC) techniques. The remaining blocks of our algorithm are standard and we provide more details in Section A of the Online Appendix. Here, we only provide an overview of the algorithm. Our sampler cycles between the following steps:

- Both the linear coefficients $\boldsymbol{\gamma}$ of dimension K and the weights vector of the output layer $\boldsymbol{W}_{L+1}$ of dimension Q_L, associated with the neurons, are obtained jointly from a standard multivariate Gaussian posterior, see Eqs. (A.1) and (A.2) in the Online Appendix.
- For shrinking the linear coefficients, we use the horseshoe prior (Carvalho et al., 2010) and update the corresponding hyperparameters by sampling from inverse Gamma distributions using the auxiliary sampler proposed in Makalic and Schmidt (2015), see Eqs. (A.3) to (A.6).
- Sampling from $p(\boldsymbol{W}_\ell|\bullet)$, for $\ell = 1, \ldots, L$, is achieved through an HMC step, see Eqs. (A.7) to (A.11).
- The shrinkage hyperparameters of the horseshoe prior on $\boldsymbol{W}_\ell$ are obtained from simple inverse Gamma posteriors, see Eqs. (A.12) to (A.15).
- The function $h_{\ell,q}$ is simulated by first introducing an indicator $\delta_{\ell,q}$ that takes integer values one to four to determine the activation function chosen. This indicator is simulated from a multinomial distribution, see Eq. (A.16).
- Draws of $\boldsymbol{v}$ and $\boldsymbol{\beta}_v$ are simulated using the algorithm proposed in Kastner and Frühwirth-Schnatter (2014).

We iterate through our MCMC algorithm 20,000 times and discard the first 10,000 draws as burn-in.

3. Illustration using synthetic data

Before we deal with actual data, it is worthwhile to investigate the properties of the different BNNs in terms of how well they recover the mean function. To do so, we need to come up with a data generating process (DGP) that closely resembles actual macroeconomic dynamics. This is done by setting up a DGP that is inspired by the model in Benigno and Eggertsson (2023), which assumes that inflation follows a nonlinear Phillips curve of the form:

$$\begin{aligned}
\pi_t &= \mu_\pi + \rho_\pi \pi_{t-1} + \{\beta_\pi + \beta_{\pi_d} D_t(\vartheta_t \geq 1)\} \log \vartheta_t + \rho_\upsilon \upsilon_t + \sigma_\pi \epsilon_{\pi,t},\\
\log \vartheta_t &= \mu_\vartheta + \rho_{\vartheta,1} \log \vartheta_{t-1} + \rho_{\vartheta,2} \log \vartheta_{t-2} + \sigma_\vartheta \epsilon_{\vartheta,t},\\
\upsilon_t &= \sigma_\upsilon \epsilon_{\upsilon,t},\\
\epsilon_{j,t} &\sim \mathcal{N}(0,1), \qquad \text{for } j \in \{\pi, \vartheta, \upsilon\},
\end{aligned}$$

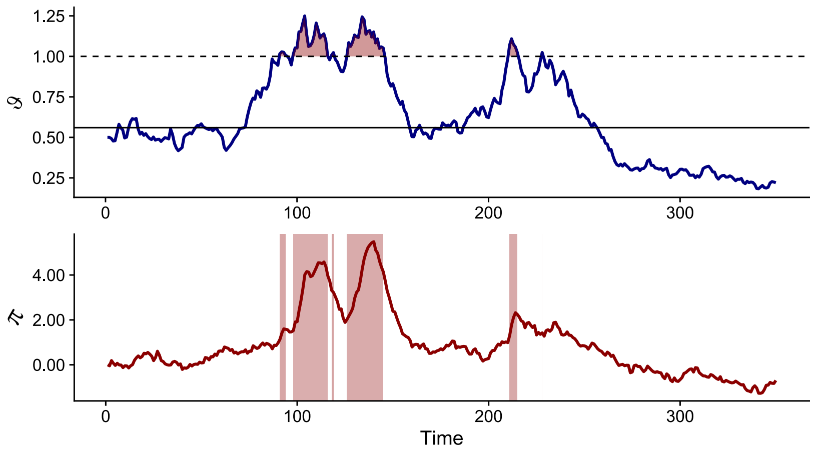


Fig. 2. A single realization from the Benigno and Eggertsson (2023) DGP. *Note:* The upper panel presents the measure of labor market tightness, ϑ_t, while the lower panel shows the resulting inflation series implied by the nonlinear Phillips curve (π_t) for a single realization from the Benigno and Eggertsson (2023) DGP. We indicate $\vartheta_t > 1$ by the red shaded area. The dashed line indicates $\vartheta_t = 1$ and the black solid line shows $\vartheta_t = 0.56$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

where ϑ_t is a measure of labor market tightness and D_t is a dummy variable taking value one if $\vartheta_t \geq 1$. We take most parameters from the full-sample estimates of Table 1 in Benigno and Eggertsson (2023). This implies setting $\mu_\pi = 0.192, \rho_\pi = 0.562, \beta_\pi = 0.222, \beta_{\pi_d} = 3.896, \rho_v = 0.0469$. σ_π is set equal to 0.1. For the second equation, which controls the dynamic evolution of the measure of labor market tightness, we set the parameters to closely match the dynamics of actual labor market tightness as proposed in Michaillat and Saez (2022) and used by Benigno and Eggertsson (2023). This requires to set $\mu_\vartheta = -0.693, \rho_{\vartheta,1} = 1.3, \rho_{\vartheta,2} = -0.3$ and $\sigma_\vartheta = 0.05$. The third equation reflects a series of supply shocks (v_t) with $\sigma_v = 1$. From this DGP, we simulate 20 time series with length $T = 350$.

This DGP is nonlinear and resembles a threshold model. In particular, as noted, if ϑ_t exceeds unity, then the slope of the labor market tightness indicator switches. The variance of the shocks to $\log \vartheta_t$ controls how often this happens in-sample. A realization from the DGP is depicted in Fig. 2. In terms of time series dynamics, it matches actual US year-on-year inflation quite well. In around 15 percent of the time, ϑ_t exceeds one, so nonlinearities kick in.

For this realization, we estimate BNNs that differ by the number of neurons per layer $Q_\ell = Q \in \{K, \dots, 15\}$, number of layers $L \in \{1, 2, 3\}$, and how we decide on the activation functions. The number of covariates is $K = 3$ and consists of the first lag of π_t, $\log \vartheta_t$ and v_t. We consider a neuron-specific mixture activation function for the BNN with a single hidden layer and a layer-specific (but not neuron-specific) mixture for cases $L > 1$. Then, we consider a BNN that includes the same activation function across all layers and neurons but we estimate it. And finally, we include a model with a ReLU activation function.

In what follows, we adopt the following labeling convention for the different models. The activation function is denoted by `flex` for a neuron or layer-specific mixture activation function, `common` for a common mixture activation function across layers and neurons, and `ReLU` for the ReLU activation function. `NN-flex` is the BNN that has activation functions that are neuron-specific for the case $L = 1$, while for $L > 1$ we estimate a single activation function per layer. `NN-ReLU` is the BNN with ReLU activation function while `NN-common` uses a common activation function for all layers and neurons.

Fig. 3 presents the average across 20 replications of in-sample relative root mean squared errors (RMSEs) from various BNNs benchmarked against the simplest BNN considered (i.e., `NN-common` with a single hidden layer and three neurons). Results are summarized in three heatmaps that illustrate the relationship between the number of neurons and layers for different specifications of the activation functions. The left panel shows the results for `NN-common`. The middle panel includes the results for the BNN that has neuron-/layer-specific activation functions (`NN-flex`) and the right panel includes `NN-ReLU`. This allows us to investigate how increasing network complexity (in terms of width, i.e. the number of neurons, and depth, i.e. the number of layers) impacts in-sample estimation accuracy.

From Fig. 3, we can draw three main conclusions. First, as opposed to other fields, estimation accuracy does not increase when we move from one to two hidden layers. This holds for `NN-common` and `NN-flex`, whereas for `NN-ReLU` the performance remains broadly unchanged. When we move from two to three layers, in all cases estimation accuracy further deteriorates. This indicates that for a time series such as the one shown in Fig. 2, using shallow networks is sufficient and, in fact, recovers implied mean relations with (slightly) more accuracy than if we include more layers.

Second, when we increase the number of neurons we find improvements in estimation accuracy for all models and most choices of Q. This, however, only holds for the first hidden layer when we consider `NN-common` and `NN-ReLU`. For `NN-flex`, it also holds for two hidden layers. Notice that for `NN-flex`, we find that after including six neurons, performance gains vanish and for $Q \geq 6$, almost all models produce a similar in-sample fit.

Finally, the single best performing model in this exercise is the `NN-flex`, with one layer and $Q \in \{10, 11, 13\}$. Given that in-sample RMSEs do not change appreciably when we include more neurons, using `NN-flex` with $Q = 15$ is a competitive choice that reduces the risk of model mis-specification. Yet, it can increase the risk of overfitting, an issue that we consider in more details in the next section.

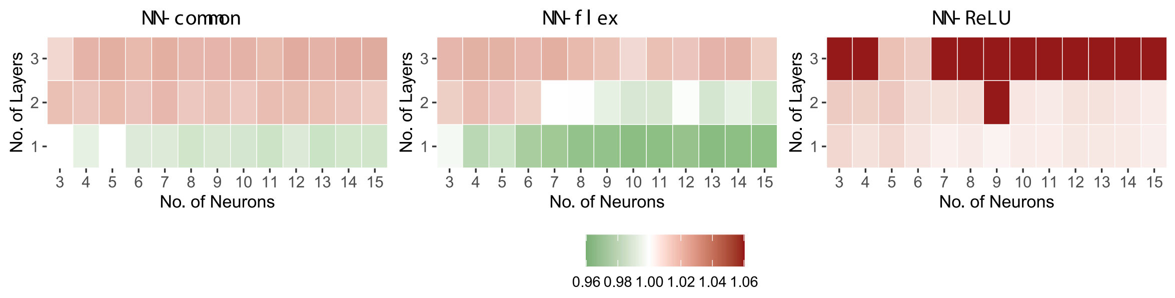


Fig. 3. Insample fit across layers and neurons. *Note:* This figure shows relative root mean squared errors (RMSEs) benchmarked against the simplest BNN considered (i.e., `NN-common` with a single hidden layer and three neurons). Values below 1 show outperformance of the benchmark (in green) while inferior specifications give values above 1 (indicated in red). We take averages over 20 random draws from our DGP. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4. Macroeconomic forecasting using BNNs

In this section, we show that different BNN models produce accurate forecasts and we explain why they do so. In the next sub-section we discuss the data, forecast design and competing models while in Section 4.2 we include the forecasting results, and in Section 4.3 we zoom into the predictive model properties.

4.1. Data overview, competing models, and the forecast exercise design

We use the popular FRED-MD database proposed in McCracken and Ng (2016). To gain a comprehensive picture of the importance of nonlinearities in large macro datasets, our forecasting exercise focuses on the consumer price (CPIAUCSL) inflation rate as specified in Stock and Watson (1999), the month-on-month (m-o-m) growth rate of industrial production (INDPRO), and the m-o-m growth rate of employment (CE16OV). The sample ranges from January 1960 to December 2020. We assume that these three focus variables are an (unknown) function of the first lag of $K = 120$ economic and financial variables. To get an understanding on the relationship between model size and forecast performance, we also consider smaller models that only include as regressor the first lag of the dependent variable and models that include the first eight principal components of the (lagged) variables.[4] These models can be interpreted as (possibly nonlinear extensions of the) diffusion index regressions in the spirit of Stock and Watson (2002).

Our focus is on short-term forecasting. Hence, we compute the one-month-ahead predictive distributions for our hold-out sample, which starts in January 2000 and ends in December 2020 (i.e., 252 monthly hold-out periods). These forecasts are obtained recursively, meaning that we use the data through January 2000 as a training sample and then forecast one-month-ahead. After obtaining the corresponding predictive densities, the sample is expanded by a single month. This procedure is repeated until the end of the sample is reached.

We consider different versions of our BNN models. They differ in terms of the number of hidden layers and the choice of the activation function. We consider a deep BNN with $L = 3$ hidden layers. Within each layer, we set the number of neurons equal to K, and thus include a large number of neurons. Then, we also consider a shallow neural network that includes only a single hidden layer. Within both types of BNNs we consider different versions of our mixture activation function. First, for shallow BNNs we set them to be neuron-specific. For deep BNNs, we estimate the activation function for each layer but not across neurons. Moreover, we consider a deep and a shallow BNN that only uses a single activation function across all layers and neurons that we estimate. The final version is a deep and shallow BNN with ReLU activation function. We state the depth of the network (which can either be `shallow` or `deep`) and then the specification for the activation function. For instance, `deep-NN-flex` is the deep neural network with layer-specific activation functions while `shallow-NN-ReLU` is the shallow BNN with ReLU activation function.

We compare the different BNNs to other techniques commonly used in machine learning and applied macroeconometrics. A natural benchmark is the linear regression model. We include several linear models that differ in how regularization is achieved. First, we consider a regression model that features a horseshoe prior. Second, we use the unobserved components model with stochastic volatility (UC-SV) of Stock and Watson (2007). Then, we include models that use LASSO (Tibshirani, 1996) and elastic net regularization (Zou and Hastie, 2005), respectively. In addition, we use Bayesian additive regression trees (BART, see e.g. Chipman et al. (2010) and Sub-section B.2 in the Online Appendix) and random forests (RF, Breiman, 2001). Last, we also consider a BNN estimated through back-propagation (labeled BNN-BP). This BNN includes the ReLU activation function and a spike and slab prior on the weights (see Sub-section B.1 in the Online Appendix for more details). Except for BNN-BP, RF and the elastic net, all models feature SV.

[4] To determine the number of factors, we follow Bai and Ng (2002, 2013) and use the IC_2 criterion.

Table 2
Forecast performance (January 2000 to December 2020).

Covariates	Model											
	deep-NN-			shallow-NN-			BNN-	BART	Elastic	LASSO	Random	UC-
	common	flex	ReLU	common	flex	ReLU	BP		net		forest	SV
Inflation												
AR(1)	**−0.01**	−0.02	−0.01	−0.02	−0.01	−0.02	−0.19	−0.22	−0.04	−0.04	−3.63	
	1.05	1.05	1.05	1.05	1.05	1.05	**1.03**	1.11	1.05	1.05	1.28	
PCA	0.01	**0.01**	0.01	0.01	0.01	0.01	−0.11	−0.20	−0.30	−0.01	−0.22	
	1.00	1.00	**1.00**	1.01	1.00	1.00	1.04	1.07	1.06	1.00	1.06	
Large	0.09**	**0.09****	0.09**	0.09**	0.09**	0.08**	−0.12	−0.03	−0.21	0.06**	−0.01	
	0.94**	0.94**	**0.93****	0.94**	0.94**	0.94**	1.03	0.97	1.00	0.95**	1.01	
UC												−0.04
												1.04
Industrial production												
AR(1)	**0.08****	0.08**	0.08**	0.08**	0.08**	0.08**	−0.81	−0.12	0.03	0.03	−2.02	
	0.91	0.91	0.90	0.91	0.91	0.91	0.92	**0.89**	0.91	0.91	0.95	
PCA	0.06**	0.06**	0.06**	0.05**	0.05**	**0.06****	−0.46	−0.01	−2.55	0.00	−0.19	
	1.00	1.01	1.00	1.00	1.01	0.99**	0.94	0.89	1.20	1.00	**0.83**	
Large	0.16**	0.16**	**0.17****	0.17**	0.16**	0.16**	−0.41	0.08	−1.75	0.09**	−0.07	
	0.98	0.98	0.97	0.97*	0.97	0.97	0.93	0.88	0.99	1.01	**0.86**	
UC												0.09**
												0.90
Employment												
AR(1)	0.26	0.24	0.26	0.26	**0.27**	0.27	−0.96	−0.67	0.08	0.08	−4.83	
	1.01	1.01	1.01	1.01	1.01	1.01	1.01	1.14	1.02	1.02	1.18	
PCA	0.27	**0.34**	0.30	0.29	0.32	0.30	−0.59	−0.19	−6.28	0.08	−3.35	
	0.99*	0.99*	1.00	0.99*	0.99	0.99*	1.01	1.03	1.00	1.01	**0.99**	
Large	0.36*	**0.41***	0.36*	0.36*	0.38*	0.36*	−0.87	0.11	−5.56	0.18	−2.49	
	1.01	1.01	1.01	1.01	1.01	1.02	1.01	1.04	**0.98***	1.00	0.99*	
UC												0.28
												1.01

Note: The table shows average log predictive likelihoods (LPLs) and root mean squared forecast errors (RMSE, gray shaded rows) relative to the linear benchmark (a diffusion index regression with SV and a horseshoe prior). In bold we mark the best performing model for each case. Asterisks indicate statistical significance by means of the one-sided Diebold and Mariano (1995) test for each model relative to the benchmark at the 5% (**) and 10% (*) significance levels. Results are averaged across the hold-out.

To compare models, we rely on log predictive likelihoods (LPLs) and root mean squared forecast errors (RMSEs). We predominantly focus on LPLs given its close relationship to the marginal likelihood, a standard Bayesian measure of model fit (see, e.g., Geweke and Amisano, 2010) and the fact that researchers in policy institutions are often interested in predicting the whole density of a target variable. Moreover, focusing on point forecast accuracy exclusively, implies that we ignore higher-order features of the predictive distributions.

4.2. *Out-of-sample predictive accuracy*

4.2.1. *Overall forecasting performance*

To gain an overall picture of the forecasting performance, we show differences in average LPLs and ratios of RMSEs (gray shaded rows), over the hold-out sample, between a particular model and the linear diffusion index regression model in Table 2. Numbers above zero (smaller than one) point towards outperformance of a given model whereas numbers below zero (greater than one) suggest that the benchmark linear model produces more accurate density predictions. One and two asterisks indicate that the differences in forecast accuracy are significant according to the one-sided Diebold and Mariano (1995) test at the ten and five percent significance level, respectively.

Starting with inflation forecasts, we observe differences across information sets. In general, using only the first lag of inflation yields density forecasts that are very close to the ones of the linear benchmark model that leverages a larger information set. Within the class of BNNs, there are only small differences across specifications. When we add the first eight principal components to the lagged inflation (the row labeled PCA in the table) we find small but insignificant gains vis-á-vis the benchmark for the BNNs. The other machine learning methods are often weaker than the different BNNs, with LASSO the best performing among them. This is not surprising given that the PCA regression is estimated under a horseshoe prior, and both the horseshoe and LASSO are related, in the sense that the LASSO can be interpreted as a Bayesian global–local shrinkage prior. Interestingly, we also find that BNNs improve upon the UC-SV model.

If we include all variables in an unrestricted manner, results change somewhat. The different BNNs outperform the benchmark. These improvements in density forecast performance are also often statistically significant at the five percent level. We find only small differences among the BNNs, with deep and shallow models producing very similar density forecasts. When we compare different specifications for the activation functions, we find no discernible differences (except for a slightly weaker performance for the shallow BNN with ReLU activation functions). In terms of the competing models, we find LASSO-based density predictions to remain competitive, whereas those from the other models are inferior to the benchmark.

Focusing on point forecasts of inflation reveals that models that do well in terms of LPLs also perform well in terms of RMSEs. Similarly to density forecasts, we find that point inflation forecasts arising from different NNs that leverage a large information set are most precise, with statistically significant improvements relative to the benchmark reaching around seven percentage points.

Next, we consider density forecasts for industrial production (IP) growth. Starting again with the AR(1) information set, we find that all BNN-based models produce LPLs that are approximately the same, and outperform the benchmark in a significant manner. The fact that predictive accuracy is very similar across BNNs is not surprising, since we only leverage information embodied in lagged IP growth. Most other models do not improve upon the benchmark model, except for the LASSO (and elastic net, which however does worse with a larger information set) and the UC-SV model. When we add more information, we find a similar pattern, with slightly larger gains for the BNNs that are rather similar across specifications.

In terms of RMSEs, a slightly different pattern shows up. BNNs which include only the first lag of IP growth improve over the benchmark. For larger information sets, these gains vanish and the neural networks are outperformed by other models such as random forests, BART, or the UC-SV. When viewed in combination with the density forecast performance, this finding indicates that gains in terms of LPLs do not arise from better point predictions but rather from higher-order features of the predictive distributions.

Finally, we consider employment growth forecasts. In terms of LPLs, neural networks, across information sets, perform well and improve upon all competing machine learning specifications by appreciable margins. Interestingly, we find more variation with respect to how we treat the activation function. In principle, if we use our mixture specification and make the activation function either layer or neuron-specific (in the case of the shallow learners), we almost always gain relative to the model that fixes the activation functions to be of ReLU type. For point forecasts, we find almost no improvements relative to the linear benchmark. This holds for most machine learning techniques in the set of benchmarks and provides evidence that, similarly to IP growth, gains in LPLs do not arise from superior point forecasts.

To sum up, different variants of the BNNs we propose outperform most other common machine learning models in terms of quality of short-term density forecasts for key US economic indicators. The gains decline when we focus on point forecasts, but the BNN point forecasts are generally at least as good as those from common benchmarks. Differences between deep and shallow BNNs are rather small, indicating that for practitioners interested in predicting output, inflation or employment, a shallow BNN that is easier to handle is a good choice that delivers accurate density forecasts on average. In fact, the Diebold and Mariano (1995) tests suggest that when NNs improve upon the benchmark, this is statistically significant for all of them. Moreover, while our mixture activation function only translates into modest gains in terms of density forecasting performance, it is worth stressing that it frees the researcher from the necessity to decide on one particular activation function and thus reduces the number of inputs to the model. We should also stress that, in most cases, using more information in an unrestricted manner pays off. This finding is not specific to all models we consider but mostly to BNNs.

4.2.2. *Forecasting performance over time*

The discussion in the previous section focused on how well the different models perform in terms of overall forecast accuracy for the full hold-out period. To drill into performance differences over time, Fig. 4 shows average one-step-ahead LPLs and rolling RMSEs relative to the linear benchmark model over time. Given their strong performance, we focus on the models that feature the large dataset and include the best performing approach among the competitors.

At a general level, we find that modeling nonlinearities pays off for density forecast performance during turbulent times such as the recession in the early 2000s, the GFC and during the pandemic. Only early in the GFC, the linear benchmark produces slightly more accurate density forecasts.[5] But halfway through the GFC this pattern shifts and most nonlinear learners outperform linear models. In terms of point forecasts, this pattern is less pronounced. While we do find improvements in point forecast accuracy during the GFC, predictive performance slightly deteriorates during the pandemic.

For density forecasts of IP and employment growth we find substantial gains of nonlinear models during recessionary episodes. Notice that for these two focus variables, we have added separate panels that show the performance throughout the pandemic. Focusing on these we find heavy gains in early 2020. These improvements persist but flatten out afterwards. This pattern, however, does not carry over to point forecast accuracy. Comparing average LPLs with rolling RMSEs shows that the strong performance in LPLs seems to be driven by superior variance predictions whereas point forecasts are similar to the one of the linear benchmark model. Such large gains in forecast performance are not evident for inflation, which remained rather stable relative to industrial production and employment growth in 2020 and did not exhibit these severe outliers.

This observation provides evidence that a high degree of model flexibility to capture severe outliers in real economic activity pays off in terms of density forecast accuracy. The comparison between point and density forecasts shows that (particularly during the pandemic) the improvements in LPLs are driven by increases in predictive variance (and thus a higher probability of observing outlying observations) and not by better median predictions.

Consistent with the overall findings, deep and shallow models display a similar performance, making it difficult to find systematic differences over time. Yet, ReLU-based models are consistently weaker than their counterparts with a mixture activation function

[5] Notice that this does not hold for the LASSO. We conjecture that this is driven by the fact that the LASSO tends to overshrink significant signals.

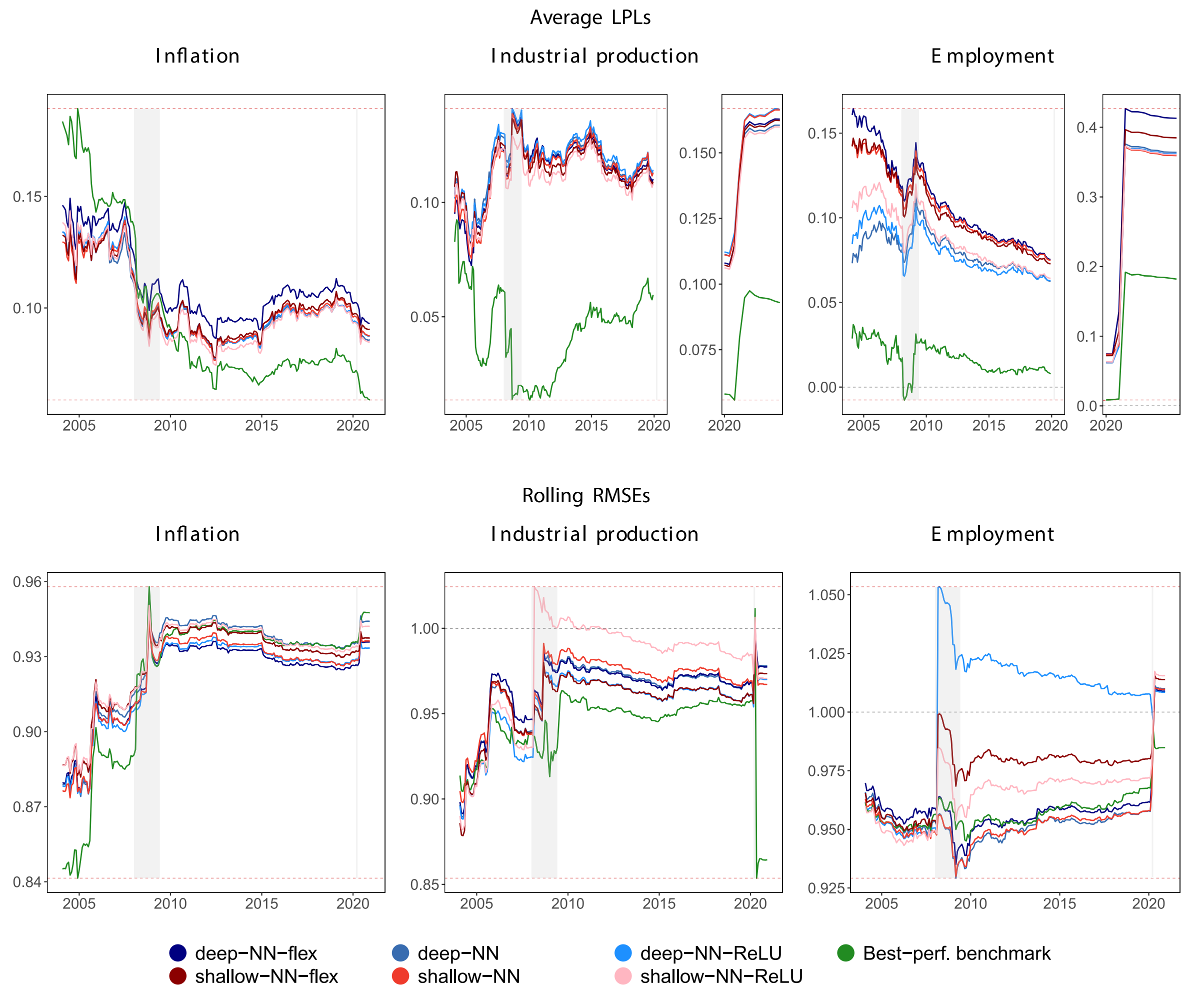


Fig. 4. Average LPLs and RMSEs for BNNs and the best performing benchmark. *Note:* We plot the evolution of average log predictive likelihoods (LPLs) and rolling root mean squared forecast errors (RMSEs) against the linear diffusion index model for the one-step-ahead predictive densities. To avoid overrepresentation of the initial root mean squared forecast errors at the beginning of the hold-out we start depicting the rolling RMSEs after the first 60 observations. The red dashed lines denote the max./min. LPLs, while the gray shaded areas indicate the NBER recessions. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

for employment. It is also interesting that deep BNNs with ReLU activation functions are beaten by shallow BNNs with flexible activation functions. This pattern holds for all three focus variables and seems to be consistent over time.[6]

Finally, to analyze specifically the role of the pandemic, we have also re-computed the forecasting results with data only through 2019:M12, with details provided in the appendix. It turns out that NNs are still highly competitive and improve upon all competing models. The only substantial change relative to our baseline results in the previous subsection is that LPL differences for employment are slightly less pronounced.

4.2.3. *The role of stochastic volatility*

One of our additions to the standard NN toolkit is the introduction of SV. To investigate the empirical relevance of this, we now re-do the forecast exercise but turn off SV for the BNNs. The results of this exercise are shown in Table 3. The table again shows the LPLs and RMSEs relative to the linear diffusion index model with SV. Hence, differences between Tables 2 and 3 are directly comparable.

Turning off SV hurts forecasting performance, in particular if the full predictive density is evaluated. In most cases, LPL differences are lower than the ones observed in Table 2 by varying margins. This also holds for RMSE ratios which are either often similar or

[6] Figure C.1 in the Online Appendix reports the fluctuation test statistic for density forecasts relative to the benchmark as proposed by Giacomini and Rossi (2010). It turns out that the gains with respect to the benchmark are present and statistically significant for all the BNN models, particularly so for inflation and industrial production, while they are only occasionally significant for LASSO.

Table 3
Forecasting performance of the various NNs without SV in the shocks.

Covariates	Model					
	deep-NN-			shallow-NN-		
	common	flex	ReLU	common	flex	ReLU
Inflation						
AR(1)	−0.26	**−0.26**	−0.26	−0.26	−0.26	−0.26
	1.05	**1.05**	1.05	1.06	1.06	1.05
PCA	−0.21	**−0.13**	−0.21	−0.21	−0.21	−0.20
	1.02	**1.02**	1.03	1.02	1.04	1.02
Large	−0.07	**−0.04**	−0.06	−0.07	−0.06	−0.07
	0.99	0.98	0.98	0.99	**0.98**	0.99
Industrial production						
AR(1)	−0.88	−0.88	**−0.84**	−0.89	−0.99	−0.89
	0.94	**0.93**	0.93	0.98	1.04	0.95
PCA	−1.66	**−0.07**	−2.06	−1.92	−2.35	−1.38
	1.26	1.28	1.33	1.33	1.40	**1.24**
Large	−0.73	**0.03**	−0.37	−0.54	−0.58	−0.63
	1.02	**1.01**	1.01	1.04	1.03	1.01
Employment						
AR(1)	−4.97	−4.72	**−4.56**	−4.60	−16.11	−5.26
	1.07	1.07	1.06	**1.06**	1.75	1.12
PCA	−6.56	**−2.42**	−5.10	−6.82	−9.87	−9.28
	1.14	1.13	**0.99**	1.14	1.33	1.29
Large	−5.93	**0.28**	−5.88	−5.29	−5.83	−6.05
	1.03	**1.02**	1.03	1.03	1.03	1.03

Note: The table shows average log predictive likelihoods (LPLs) and root mean squared forecast errors (RMSE, gray shaded rows) relative to the linear benchmark (a diffusion index regression with SV and a horseshoe prior). In bold we mark the best performing model for each case. Asterisks indicate statistical significance by means of the one-sided Diebold and Mariano (1995) test for each model relative to the benchmark at the 5% (**) and 10% (*) significance levels. Results are averaged across the hold-out.

worse than the ones of the heteroskedastic NNs. In fact, for inflation, we find that the benchmark linear model with SV improves upon `deep-NN-flex` and `shallow-NN-flex` with the large dataset. There are only small differences in terms of forecast accuracy between the deep and shallow models. This is consistent with the findings based on the models that use SV.

For IP and employment growth, we find that `deep-NN-flex` is capable of improving upon the benchmark. These gains are muted for IP and slightly more pronounced for employment. What is interesting, however, is that the choice of the activation function seems to matter much more when we use homoskedastic BNNs. The layer-specific mixture activation function in the case of a deep BNN yields much more precise density forecasts than a ReLU-based BNN and a specification with a common mixture activation function. Notice, that for IP growth we observe that deep BNNs seem to cope much better with model mis-specification in terms of the shock volatility processes than shallow models.

4.3. Predictive model properties

To better understand the predictive performance of our different BNNs, we now consider two specifications and zoom into specific model features. The first is the deep BNN with layer-specific activation functions (`deep-NN-flex`) and the second one is the shallow BNN with neuron-specific activation functions (`shallow-NN-flex`).

4.3.1. Which activation function?

A natural starting point is the specific form of the activation function if we use our mixture specification. To investigate which activation function receives more model weight, we focus on how much weight each activation function attains under the posterior. Recall that the mixture activation function mixes over four different activation functions (leakyReLU, sigmoid, ReLU, and tanh). Table 4 shows the average of the posterior estimates of $\omega_{\ell,q}^{(m)}$ associated with each of the four activation functions over the hold-out period. The rows show the different layers and, to simplify the table, we have averaged over the weights associated with the different neurons in the case of `shallow-NN-flex`.

For the shallow model, we find that all activation functions receive almost equal weights across the three focus variables. Only sigmoid and tanh obtain slightly more posterior weight. This feature is more pronounced for employment, but differences are quite small.

For `deep-NN-flex` more interesting patterns arise. In this case, we find more asymmetries in terms of activation functions across variables and layers. For inflation, tanh receives slightly more posterior weight for the first two hidden layers. In the output layer, this pattern shifts and leakyReLU and ReLU attain only little weight while sigmoid and tanh obtain over 90 percent of total

Table 4
Posterior weights of activation functions.

Layer	Inflation				Industrial production				Employment			
	leakyReLU	ReLU	sigmoid	tanh	leakyReLU	ReLU	sigmoid	tanh	leakyReLU	ReLU	sigmoid	tanh
`shallow-NN-flex`												
1	23.9	23.8	26.2	26.1	23.8	23.8	26.3	26.1	23.7	23.7	26.0	26.6
`deep-NN-flex`												
1	24.6	24.0	23.4	28.0	24.9	23.8	23.1	28.2	21.2	19.9	19.4	39.5
2	24.2	24.1	23.6	28.1	24.1	23.9	24.9	27.1	25.8	21.2	21.0	32.0
3	3.5	3.4	44.3	48.8	3.6	3.6	45.2	47.6	3.6	3.6	38.3	54.5

Note: The table shows mean posterior estimates of $\omega_{\ell,q}^{(m)}$ for each layer averaged across the hold-out periods. For `shallow-NN-flex`, we average over the weights of all neurons. All numbers are in percentages.

posterior weight. For IP, a similar pattern shows up. In the first two layers, we find posterior weights that are more symmetric while in the output layer, sigmoid and tanh dominate and leakyReLU and ReLU only play a limited role.

For employment growth, the variable for which we find the strongest gains, the pattern is different. For layers one and two, tanh receives the largest posterior weight with the remaining three activation functions having similar mean estimates of $\omega_{\ell,q}^{(m)}$. For the output layer, we again find that sigmoid becomes substantially more important, receiving a weight of around 38 percent while tanh still dominates and receives over 54 percent of posterior weights.

This brief discussion shows that in the predictive exercise, our BNNs do not generate strong forecasts by relying on a single activation function but by averaging over all four of them (in particular for the deep model and hidden layers below the output layer). In the case of the deep BNN, the model places substantial posterior weight on sigmoid and tanh, a pattern that also shows up for the shallow model but in a much more attenuated manner.

4.3.2. The relationship between in-sample fit and out-of-sample predictability

The results up to this point tell a story that flexible models help in extreme periods and are competitive in normal times. These are much more pronounced for density forecasts while the improvements for point forecasts are more muted. To gain a better understanding on how flexible NNs improve the estimation of the conditional mean and whether this yields superior density forecasts, we investigate the relationship between in-sample fit and out-of-sample predictive capabilities. This exercise allows us to answer the question whether NNs can extract information from $\boldsymbol{x}_t$ that linear models cannot exploit and how this impacts predictive accuracy.

To achieve this, we compute the amount of variation explained through the conditional mean piece (labeled R2) for `shallow-NN-flex` and `deep-NN-flex`. These R2s are computed recursively and put in relation to the corresponding t-by-t LPL using a simple scatter plot. The scatter plots are provided in Fig. 5. The horizontal line at zero implies that if a point is below zero, the linear regression model produces superior density forecasts whereas in the opposite case, the BNN is forecasting better. Points to the left of the vertical line (which stands at one) imply less explanatory power of the BNN whereas points to the right indicate that the conditional mean part of the BNN explains more of the variation in the response as the linear model.

Two examples illustrate how the scatter plot can be interpreted. Points in the quadrant with R2 > 1 and LPL > 0 represent situations where the BNN is extracting information from $\boldsymbol{x}_t$ that leads to a higher in-sample fit and this information translates into more accurate density forecasts. If R2 ≈ 1 but LPL > 0 both models explain a similar amount of in-sample variation but density forecasting performance of the BNN is superior. In this case, these differences are likely driven by higher order moments of the predictive distribution.

Starting with inflation (see panel (a) of the figure) reveals two interesting differences across shallow and deep learners. The shallow neural network produces competitive forecasts based on an in-sample fit that is comparable to the linear diffusion index regression whereas `deep-NN-flex` produces density forecasts of similar quality but does so with a much stronger in-sample fit that is almost always higher than the one of the linear model, with relative R2 frequently exceeding 1.5.

For IP growth (depicted in panel (b)), we observe a similar pattern. Shallow models produce in-sample fits which are close to the ones of the linear model. By contrast, the R2-LPL combinations for the deep BNNs are mostly located in the top right quadrant. In both cases, however, BNNs produce IP forecasts in the pandemic (April 2020) which are much more precise than the ones of the linear model and do so while explaining more of the in-sample fit. For `deep-NN-flex` we also find that density predictions for output growth are much more precise during September 2008, the month of the collapse of Lehman Brothers.

The overall story that shallow BNNs produce accurate density forecasts with an in-sample fit that is close to the one of the linear model while deep models do the same but with a much higher in-sample fit carries over to employment growth predictions in panel (c). For shallow BNNs we find that around half of the points are above one in terms of relative R2 and for deep BNNs this share is much higher. In both cases, nonlinear models heavily outperform the linear benchmark in April 2020 and, to a lesser degree, in March 2020. In both periods, employment numbers exhibited historic declines due to the Covid-19 pandemic.

Table 5 summarizes these findings in a few numbers. The first row, labeled 'Outperformance' shows the number of times a particular BNN improves upon the linear benchmark, the second row, labeled 'Outperformance & higher R2' gives the percentage of times a model improves upon the linear model and does so with a higher in-sample R2. This table shows that shallow models have a

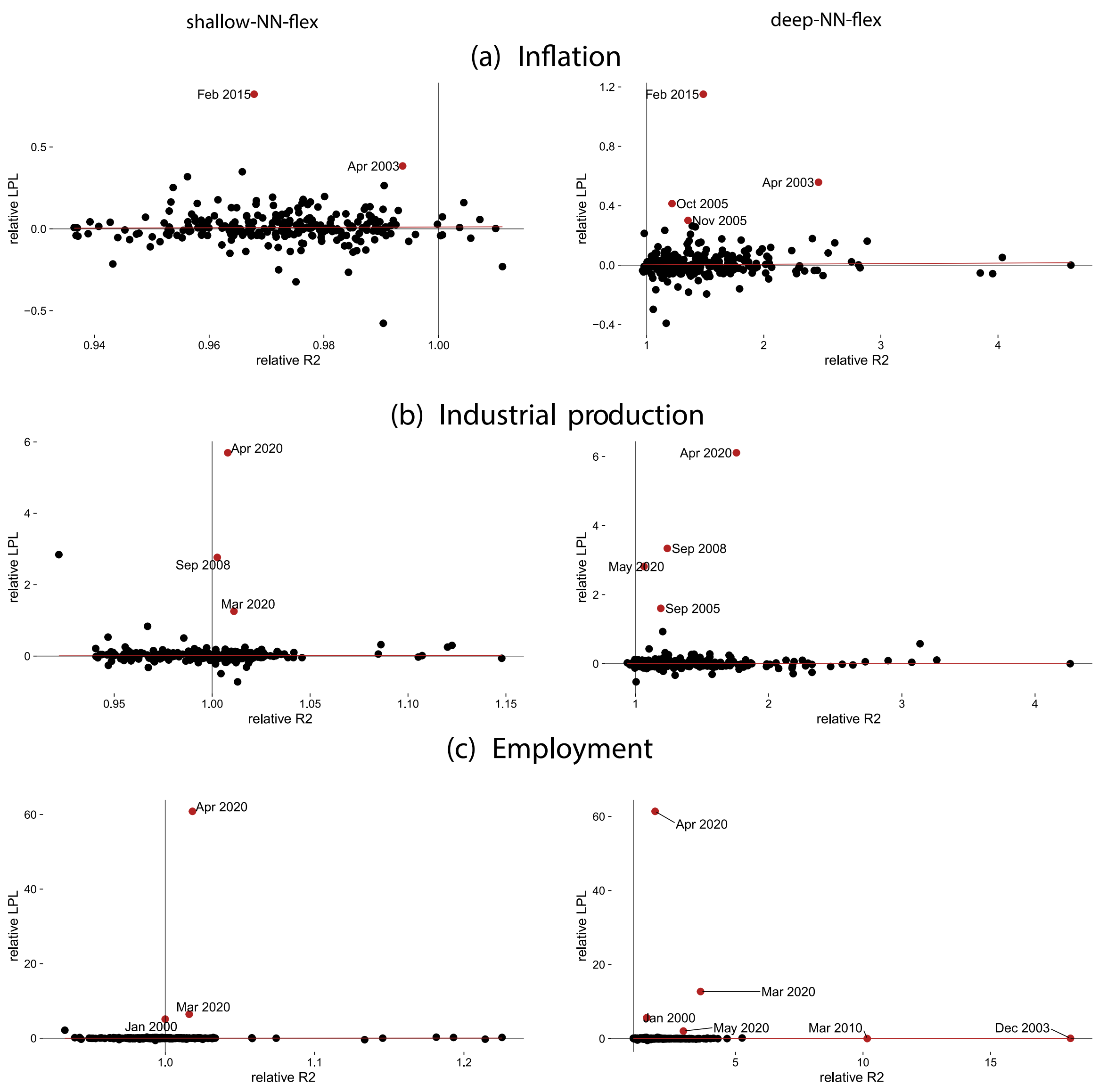


Fig. 5. Relative R2 against relative LPL. *Note:* This figure shows relative R2 versus relative LPL of `shallow-NN-flex` and `deep-NN-flex` against the linear model for each of our three variables. We color observations which feature high in-sample fit and high predictive accuracy relative to the benchmark. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 5
Measures for the in-sample and out-of-sample outperformance of the BNN against the linear benchmark.

	Inflation		Industrial Production		Employment	
	shallow-NN-flex	deep-NN-flex	shallow-NN-flex	deep-NN-flex	shallow-NN-flex	deep-NN-flex
Outperformance	54.0	48.8	54.0	46.4	55.6	53.6
Outperformance & higher R2	2.0	48.4	26.6	45.2	25.0	53.2

Note: The table shows two different measures describing the relationship between in-sample fit and out-of-sample predictability. The measure "Outperformance" gives the share of relative LPL being positive (i.e., corresponding to the datapoints in the upper quadrants of Fig. 5). The second measure "Outperformance & higher R2" gives the share of relative LPL being positive and a relative R2 above one (i.e., corresponding to the datapoints in the right-upper quadrants of Fig. 5). All numbers are in percentages.

slightly higher share of outperformance than deep models. For `shallow-NN-flex`, this outperformance is often achieved with a lower R2 than the benchmark (two percent for inflation; 27 percent for IP and 25 percent for employment). By contrast, deep BNNs frequently outperform the benchmark with higher R2 measures. This pattern points towards benign overfitting, commonly found in the literature (see, e.g., Bartlett et al., 2020), but it also suggests that, as far as density forecasting inflation, industrial production and employment is considered, shallow BNNs can be preferred to deep BNNs, as differences in density forecasts are minor and the computational costs are much smaller.

5. Conclusion

Neural networks are extremely popular in many different fields. In economics and econometrics, their usage is growing but still comparatively limited. In this paper, we develop techniques to specify and estimate neural networks without requiring cross-validation or much input from the researcher. The key ingredient is a flexible mixture specification on the activation function, which frees the researcher from the necessity to pick a particular activation function. As an additional technical improvement, we allow for heteroskedasticity in the shocks. From a computational point of view, we develop an efficient and scalable (Bayesian) estimation algorithm.

We illustrate how these models can be used for informing policy decisions. In particular, we start by investigating the trade-off between the number of neurons and the number of hidden layers using synthetic data from a realistic DGP that resembles actual US inflation dynamics. We show that, as opposed to theoretical recommendations, a shallow neural network produces favorable MSE ratios and is capable of improving upon deep BNNs. We then move on to use the models to produce density forecasts of key US macroeconomic aggregates. In this exercise, we find that our BNNs improve upon models commonly used in machine learning, and shallow BNNs perform as well as deep BNNs, while the latter produce much better fit.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.jeconom.2024.105843.

References

Agostinelli, F., Hoffman, M., Sadowski, P., Baldi, P., 2014. Learning activation functions to improve deep neural networks. arXiv:1412.6830.

Ai, C., Norton, E.C., 2003. Interaction terms in logit and probit models. Econom. Lett. 80 (1), 123–129.

Amisano, G., Fagan, G., 2013. Money growth and inflation: A regime switching approach. J. Int. Money Finance 33, 118–145.

Bai, J., Ng, S., 2002. Determining the number of factors in approximate factor models. Econometrica 70 (1), 191–221.

Bai, J., Ng, S., 2013. Principal components estimation and identification of static factors. J. Econometrics 176 (1), 18–29.

Bartlett, P.L., Long, P.M., Lugosi, G., Tsigler, A., 2020. Benign overfitting in linear regression. Proc. Natl. Acad. Sci. 117 (48), 30063–30070.

Benigno, P., Eggertsson, G.B., 2023. It's Baaack: The Surge in Inflation in the 2020s and the Return of the Non-Linear Phillips Curve. Working Paper Series 31197, National Bureau of Economic Research.

Bhadra, A., Datta, J., Li, Y., Polson, N., 2020. Horseshoe regularisation for machine learning in complex and deep models. Internat. Statist. Rev. 88 (2), 302–320.

Bollerslev, T., 1986. Generalized autoregressive conditional heteroskedasticity. J. Econometrics 31 (3), 307–327.

Breiman, L., 2001. Random forests. Mach. Learn. 45, 5–32.

Carvalho, C.M., Polson, N.G., Scott, J.G., 2009. Handling sparsity via the horseshoe. In: Artificial Intelligence and Statistics. PMLR, pp. 73–80.

Carvalho, C.M., Polson, N.G., Scott, J.G., 2010. The horseshoe estimator for sparse signals. Biometrika 97 (2), 465–480.

Chipman, H.A., George, E.I., McCulloch, R.E., 2010. BART: Bayesian additive regression trees. Ann. Appl. Stat. 4 (1), 266–298.

Chronopoulos, I., Chrysikou, K., Kapetanios, G., Mitchell, J., Raftapostolos, A., 2023. Deep neural network estimation in panel data models. arXiv:2305.19921.

Chronopoulos, I.C., Raftapostolos, A., Kapetanios, G., 2024. Forecasting value-at-risk using deep neural network quantile regression. J. Financ. Econom. 22 (3), 636–669.

Clark, T.E., 2011. Real-time density forecasts from Bayesian vector autoregressions with stochastic volatility. J. Bus. Econom. Statist. 29 (3), 327–341.

Clark, T.E., Huber, F., Koop, G., Marcellino, M., Pfarrhofer, M., 2023. Tail forecasting with multivariate Bayesian additive regression trees. Internat. Econom. Rev. 64 (3), 979–1022.

Clark, T.E., Huber, F., Koop, G., Marcellino, M., Pfarrhofer, M., 2024. Investigating growth-at-risk using a multicountry non-parametric quantile factor model. J. Bus. Econom. Statist. (forthcoming).

Clark, T.E., McCracken, M.W., 2006. The predictive content of the output gap for inflation: Resolving in-sample and out-of-sample evidence. J. Money Credit Bank. 38 (5), 1127–1148.

Crawford, L., Flaxman, S.R., Runcie, D.E., West, M., 2019. Variable prioritization in nonlinear black box methods: A genetic association case study. Ann. Appl. Stat. 13 (2), 958.

Diebold, F.X., Mariano, R.S., 1995. Comparing predictive accuracy. J. Bus. Econom. Statist. 13 (3), 253–263.

Engle, R.F., Rangel, J.G., 2008. The spline-GARCH model for low-frequency volatility and its global macroeconomic causes. Rev. Financ. Stud. 21 (3), 1187–1222.

Escobar, M.D., West, M., 1995. Bayesian density estimation and inference using mixtures. J. Amer. Statist. Assoc. 90 (430), 577–588.

Farrell, M.H., Liang, T., Misra, S., 2021. Deep neural networks for estimation and inference. Econometrica 89 (1), 181–213.

Gallegati, M., 2008. Wavelet analysis of stock returns and aggregate economic activity. Comput. Statist. Data Anal. 52 (6), 3061–3074.

Geweke, J., Amisano, G., 2010. Comparing and evaluating Bayesian predictive distributions of asset returns. Int. J. Forecast. 26 (2), 216–230.

Ghosh, S., Yao, J., Doshi-Velez, F., 2019. Model selection in Bayesian neural networks via horseshoe priors. J. Mach. Learn. Res. 20 (182), 1–46.

Giacomini, R., Rossi, B., 2010. Forecast comparisons in unstable environments. J. Appl. Econometrics 25 (4), 595–620.

Goulet Coulombe, P., 2024. The macroeconomy as a random forest. J. Appl. Econometrics 39 (3), 401–421.

Goulet Coulombe, P., Frenette, M., Klieber, K., 2023. From reactive to proactive volatility modeling with hemisphere neural networks. arXiv:2311.16333.

Greene, W., 2010. Testing hypotheses about interaction terms in nonlinear models. Econom. Lett. 107 (2), 291–296.

Hamilton, J.D., 1989. A new approach to the economic analysis of nonstationary time series and the business cycle. Econometrica 57 (2), 357–384.

Hauzenberger, N., Huber, F., Marcellino, M., Petz, N., 2024. Gaussian process vector autoregressions and macroeconomic uncertainty. J. Bus. Econom. Statist. (forthcoming).

Hornik, K., Stinchcombe, M., White, H., 1989. Multilayer feedforward networks are universal approximators. Neural Netw. 2 (5), 359–366.

Huber, F., Koop, G., Onorante, L., Pfarrhofer, M., Schreiner, J., 2023. Nowcasting in a pandemic using non-parametric mixed frequency VARs. J. Econometrics 232 (1), 52–69.
Imbens, G.W., Wooldridge, J.M., 2009. Recent developments in the econometrics of program evaluation. J. Econ. Lit. 47 (1), 5–86.
Karlik, B., Olgac, A.V., 2011. Performance analysis of various activation functions in generalized MLP architectures of neural networks. Int. J. Artif. Intell. Expert Syst. 1 (4), 111–122.
Kastner, G., Frühwirth-Schnatter, S., 2014. Ancillarity-sufficiency interweaving strategy (ASIS) for boosting MCMC estimation of stochastic volatility models. Comput. Statist. Data Anal. 76, 408–423.
Kourentzes, N., 2013. Intermittent demand forecasts with neural networks. Int. J. Prod. Econ. 143 (1), 198–206.
Lara-Benítez, P., Carranza-García, M., Riquelme, J.C., 2021. An experimental review on deep learning architectures for time series forecasting. Int. J. Neural Syst. 31 (03), 2130001.
LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. Proc. IEEE 86 (11), 2278–2324.
Lee, D.S., Lemieux, T., 2010. Regression discontinuity designs in economics. J. Econ. Lit. 48 (2), 281–355.
Lim, B., Zohren, S., 2021. Time-series forecasting with deep learning: a survey. Phil. Trans. R. Soc. A 379 (2194), 20200209.
Makalic, E., Schmidt, D.F., 2015. A simple sampler for the horseshoe estimator. IEEE Signal Process. Lett. 23 (1), 179–182.
McCracken, M.W., Ng, S., 2016. FRED-MD: A monthly database for macroeconomic research. J. Bus. Econom. Statist. 34 (4), 574–589.
McCrary, J., 2008. Manipulation of the running variable in the regression discontinuity design: A density test. J. Econometrics 142 (2), 698–714.
Mhaskar, H., Liao, Q., Poggio, T., 2017. When and why are deep networks better than shallow ones? Proc. AAAI Conf. Artif. Intell. 31 (1).
Michaillat, P., Saez, E., 2022. $u^* = \sqrt{uv}$. Working Paper Series 30211, National Bureau of Economic Research.
Nalisnick, E., Hernández-Lobato, J.M., Smyth, P., 2019. Dropout as a structured shrinkage prior. In: International Conference on Machine Learning. PMLR, pp. 4712–4722.
Nguyen, T.-N., Tran, M.-N., Gunawan, D., Kohn, R., 2023. A statistical recurrent stochastic volatility model for stock markets. J. Bus. Econom. Statist. 41 (2), 414–428.
Polson, N.G., Ročková, V., 2018. Posterior concentration for sparse deep learning. Adv. Neural Inf. Process. Syst. 31.
Polson, N.G., Scott, J.G., 2010. Shrink globally, act locally: Sparse Bayesian regularization and prediction. Bayes. Stat. 9, 501–538.
Ramsey, J.B., Lampart, C., 1998. The decomposition of economic relationships by time scale using wavelets: Expenditure and income. Stud. Nonlinear Dyn. Econom. 3 (1).
Reichlin, L., Lenza, M., 2007. On Short-Term and Long-Term Causality of Money to Inflation: Understanding the Problem and Clarifying Some Conceptual Issues. Discussion Paper, Mimeo.
Salinas, D., Flunkert, V., Gasthaus, J., Januschowski, T., 2020. Deepar: Probabilistic forecasting with autoregressive recurrent networks. Int. J. Forecast. 36 (3), 1181–1191.
Schmidt-Hieber, J., 2020. Nonparametric regression using deep neural networks with ReLU activation function. Ann. Statist. 48 (4), 1875–1897.
Sezer, O., Ozbayoglu, M., Dogdu, E., 2020. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. arXiv:abs/1911.13288.
Stock, J., Watson, M., 1999. Forecasting inflation. J. Monetary Econ. 44 (2), 293–335.
Stock, J.H., Watson, M.W., 2002. Macroeconomic forecasting using diffusion indexes. J. Bus. Econom. Statist. 20 (2), 147–162.
Stock, J.H., Watson, M.W., 2007. Why has US inflation become harder to forecast? J. Money Credit Bank. 39, 3–33.
Teräsvirta, T., 1994. Specification, estimation, and evaluation of smooth transition autoregressive models. J. Amer. Statist. Assoc. 89 (425), 208–218.
Tibshirani, R., 1996. Regression shrinkage and selection via the LASSO. J. R. Stat. Soc. Ser. B Stat. Methodol. 58 (1), 267–288.
Tipping, M.E., 2001. Sparse Bayesian learning and the relevance vector machine. J. Mach. Learn. Res. 1 (Jun), 211–244.
Titterington, D.M., 2004. Bayesian methods for neural networks and related models. Stat. Sci. 128–139.
Tong, H., 1990. Non-Linear Time Series: A Dynamical System Approach. Oxford University Press.
Turner, A.J., Miller, J.F., 2014. Neuroevolution: Evolving heterogeneous artificial neural networks. Evol. Intell. 7, 135–154.
Vasicek, O.A., Fong, H.G., 1982. Term structure modeling using exponential splines. J. Finance 37 (2), 339–348.
Wang, Y., Rockova, V., 2020. Uncertainty quantification for sparse deep learning. In: International Conference on Artificial Intelligence and Statistics. PMLR, pp. 298–308.
Wen, R., Torkkola, K., Narayanaswamy, B., Madeka, D., 2017. A multi-horizon quantile recurrent forecaster. arXiv:1711.11053.
Williams, C.K., Rasmussen, C.E., 2006. Gaussian Processes for Machine Learning, vol. 2, MIT Press, Cambridge, MA.
Yao, X., 1999. Evolving artificial neural networks. Proc. IEEE 87 (9), 1423–1447.
Zou, H., Hastie, T., 2005. Regularization and variable selection via the elastic net. J. R. Stat. Soc. Ser. B Stat. Methodol. 67 (2), 301–320.