International Journal of Forecasting 39 (2023) 901–921

Contents lists available at ScienceDirect

International Journal of Forecasting

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

Real-time inflation forecasting using non-linear dimension reduction techniques☆

Niko Hauzenberger [a,b], Florian Huber [a], Karin Klieber [a,*]

[a] Department of Economics, University of Salzburg, Austria
[b] Department of Socioeconomics, Vienna University of Economics and Business, Austria

ARTICLE INFO

Keywords:
Non-linear principal components
Machine learning
Time-varying parameter regression
Density forecasting
Real-time data

ABSTRACT

In this paper, we assess whether using non-linear dimension reduction techniques pays off for forecasting inflation in real-time. Several recent methods from the machine learning literature are adopted to map a large dimensional dataset into a lower-dimensional set of latent factors. We model the relationship between inflation and the latent factors using constant and time-varying parameter (TVP) regressions with shrinkage priors. Our models are then used to forecast monthly US inflation in real-time. The results suggest that sophisticated dimension reduction methods yield inflation forecasts that are highly competitive with linear approaches based on principal components. Among the techniques considered, the Autoencoder and squared principal components yield factors that have high predictive power for one-month- and one-quarter-ahead inflation. Zooming into model performance over time reveals that controlling for non-linear relations in the data is of particular importance during recessionary episodes of the business cycle or the current COVID-19 pandemic.

1. Introduction

Inflation expectations are used as crucial inputs for economic decision-making in central banks such as the European Central Bank (ECB) and the US Federal Reserve (Fed). Given current and expected inflation, economic agents decide how much to consume, save and invest. In addition, measures of inflation expectations are often employed to estimate the slope of the Phillips curve and infer the output gap or the natural rate of interest. Hence, accurately predicting inflation is key for designing and implementing appropriate monetary policies in a forward-looking manner.

Although the literature on modeling inflation is voluminous and the efforts invested considerable, predicting inflation remains a difficult task and simple univariate models are still difficult to beat (Stock & Watson, 2007). The recent literature, however, has shown that using large datasets (Stock & Watson, 2002b) and/or sophisticated models (see Chan et al., 2018; Clark & Ravazzolo, 2015; D'Agostino et al., 2013; Jarocinski & Lenza, 2018; Koop & Korobilis, 2012, 2013; Koop & Potter, 2007) has the potential to improve upon simpler benchmarks.

These studies often exploit information from huge datasets. This is commonly achieved by extracting a relatively small number of principal components (PCs) and

☆ We would like to thank the participants of the 2021 Annual Meeting of the Austrian Economic Association (NOeG, June 2021), a joint internal seminar of the University of Salzburg, and the University of Strathclyde, and the KOF Research Seminar at the ETH Zurich, two anonymous referees as well as Gary Koop, Michael Pfarrhofer, Aristeidis Raftapostolos, and Anna Stelzer for valuable comments and suggestions. The authors gratefully acknowledge financial support from the Austrian Science Fund (FWF, grant no. ZK 35) and the Oesterreichische Nationalbank, Austria (OeNB, Anniversary Fund, project no. 18127).

* Correspondence to: Department of Economics, University of Salzburg, Mönchsberg 2a, 5020 Salzburg, Austria.
E-mail address: karin.klieber@plus.ac.at (K. Klieber).

https://doi.org/10.1016/j.ijforecast.2022.03.002

including them in a second stage regression model (see, e.g., Stock & Watson, 2002b). While this approach performs well empirically and yields consistent estimators for the latent factors, it fails to capture non-linear relations in the dataset. In the presence of non-linearities, using simple PCs potentially reduces predictive accuracy by ignoring important features of the data. Some studies deal with this issue by using flexible factor models, which allow for non-linearities in the data. Bai and Ng (2008) use targeted predictors coupled with quadratic principal components and show that allowing for non-linearities yields non-trivial improvements in predictive accuracy for inflation. This suggests that non-linearities (of a known form) are present in US macroeconomic datasets, commonly employed for inflation forecasting. More recently, Pelger and Xiong (2021) propose a flexible state-dependent factor model and apply this method to US bond yields and stock returns. This non-linear and non-parametric technique yields results that differ from linear, PC-based models by extracting significantly more information from the data.

One additional assumption commonly made is that the relationship between inflation and the latent factors is constant. This assumption is strong for longer time series that feature multiple structural breaks and may harm predictive accuracy. Several recent papers deal with this issue by using time-varying parameter (TVP) regressions which, in addition, allow for heteroscedasticity through stochastic volatility (SV) (Belmonte et al., 2014b; Clark & Ravazzolo, 2015; D'Agostino et al., 2013; Jarocinski & Lenza, 2018; Koop & Korobilis, 2012; Koop & Potter, 2007; Korobilis, 2021).

Investigating whether allowing for non-linearities in the compression stage pays off for inflation forecasting is the key objective of the present paper. Building on recent advances in machine learning (see Chakraborty & Joseph, 2017; Coulombe et al., 2019; Exterkate et al., 2016; Feng et al., 2018; Gallant & White, 1992; Heaton et al., 2017; Kelly et al., 2019; McAdam & McNelis, 2005; Medeiros et al., 2021; Mullainathan & Spiess, 2017), we adopt several non-linear dimension reduction techniques. The resulting latent factors are linked to inflation in a second stage regression. To investigate whether a relationship exists between non-linear factor estimation and flexible modeling of the predictive inflation equation, we introduce dynamic regression models that allow for TVPs and SV. Since including many latent factors can still imply a considerable number of parameters (and this problem is even more severe in the TVP regression case), we rely on state-of-the-art shrinkage techniques.

From an empirical standpoint, it is necessary to investigate how these dimension reduction techniques perform over time and during different business cycle phases. We show this by carrying out a thorough real-time forecasting experiment for the US. Our forecasting application uses monthly real-time datasets (i.e., the FRED-MD database proposed in McCracken & Ng, 2016) and includes a battery of well-established models commonly used in central banks and other policy institutions to forecast inflation. These include simple benchmarks as well as more elaborate models such as the specification proposed in Stock and Watson (2002b).

Our results show that non-linear dimension reduction techniques yield highly competitive forecasts to (and, in fact, often better than) the ones obtained from using linear methods based on PCs. In terms of one-month-ahead forecasts, we find that models based on the Autoencoder yield point and density forecasts that are more precise than the ones obtained from other sophisticated non-linear dimension reduction techniques as well as traditional methods based on PCs. When the focus is on one-quarter-ahead forecasts, we find that non-linear variants of PCs perform best. This performance, however, is not homogeneous over time and some of the models do better than others during different stages of the business cycle. In a brief discussion, we also analyze how our set of models performs during the COVID-19 pandemic.

These findings give rise to the second contribution of our paper. Since we observe that more sophisticated non-linear dimension reduction methods outperform more straightforward techniques during recessions, we combine the different models using dynamic model averaging (see Koop & Korobilis, 2013; Raftery et al., 2010). We show that combining our proposed set of models with various standard forecasting models yields predictive densities that are very close to the single best-performing model. This suggests that using model and forecasting averaging successfully controls for model uncertainty in light of the enormous set of models we consider.

The remainder of this paper is structured as follows. Section 2 discusses our proposed set of dimension reduction techniques. Section 3 introduces the econometric modeling environment that we use to forecast inflation. Section 4 first provides some in-sample features, then discusses the results of the forecasting horse race, and finally presents our findings based on forecast averaging. The last section summarizes and concludes the paper. The Online Appendix provides further details on the econometric techniques as well as the data and additional empirical results.

2. Linear and non-linear dimension reduction techniques

Suppose that we are interested in predicting inflation using a large number of K regressors that we store in a $T \times K$ matrix $\boldsymbol{X} = (\boldsymbol{x}_1, \dots, \boldsymbol{x}_T)'$, where $\boldsymbol{x}_t$ denotes a K-dimensional vector of observations at time t. If K is large relative to T, estimation of an unrestricted model that uses all columns in $\boldsymbol{X}$ quickly becomes cumbersome and overfitting issues arise. As a solution, dimension reduction techniques are commonly employed (see, e.g., Bernanke et al., 2005; Stock & Watson, 2002b). These methods strike a balance between model fit and parsimony. At a very general level, the key idea is to introduce a function f that takes the matrix $\boldsymbol{X}$ as input and yields a lower-dimensional representation $\boldsymbol{Z} = f(\boldsymbol{X}) = (\boldsymbol{z}_1, \dots, \boldsymbol{z}_T)'$, which is of dimension $T \times q$, as output. The critical assumption to achieve parsimony is that $q \ll K$. The latent factors in $\boldsymbol{Z}$ are then linked to inflation through a dynamic regression model (see Section 3).

The function $f : \mathbb{R}^{T \times K} \rightarrow \mathbb{R}^{T \times q}$ is typically assumed to be linear, with the most prominent example being PCs.

In this paper, we will consider several choices of f that range from linear to highly non-linear (such as manifold learning as well as deep learning) specifications. We subsequently analyze how these different specifications impact inflation forecasting accuracy. We briefly discuss the other techniques in the following subsections and refer to the original papers for additional information.

2.1. Principal component analysis

We start our discussion by considering principal component analysis (PCA). Minor alterations of the standard PCA approach introduce non-linearities in two ways. First, we can introduce a non-linear function g that maps the covariates onto a matrix $\boldsymbol{W} = g(\boldsymbol{X})$. Second, we could alter the sample covariance matrix (the kernel) with a function h: $\boldsymbol{\kappa} = h(\boldsymbol{W}'\boldsymbol{W})$. Both $\boldsymbol{W}$ and $\boldsymbol{\kappa}$ form the two main ingredients of a general PCA reducing the dimension to q, as outlined below (for details, see Schölkopf et al., 1998).

Independent of the functional form of g and h, we obtain PCs by performing a truncated singular value decomposition (SVD) of the transformed sample covariance matrix $\boldsymbol{\kappa}$. Conditional on the first q eigenvalues, the resulting factor matrix $\boldsymbol{Z}$ is of dimension $T \times q$. For appropriate q, these PCs explain most variation in $\boldsymbol{X}$. In the following, the relationship between the PCs and $\boldsymbol{X}$ is:

$$\boldsymbol{Z} = f(\boldsymbol{X}) = g(\boldsymbol{X})\boldsymbol{\Lambda}(\boldsymbol{\kappa}) = \boldsymbol{W}\boldsymbol{\Lambda}(\boldsymbol{\kappa}), \tag{1}$$

with $\boldsymbol{\Lambda}(\boldsymbol{\kappa})$ being the truncated $K \times q$ eigenvector matrix of $\boldsymbol{\kappa}$ (Stock & Watson, 2002b). Notice that this is always conditional on deciding on a suitable number q of PCs. The number of factors is a crucial parameter that strongly influences predictive accuracy and inference (Bai & Ng, 2002). In our empirical work, we consider a small ($q = 5$), a moderate ($q = 15$), and a large ($q = 30$) number of PCs.

By varying the functional form of g and h, we are now able to discuss the first set of linear and non-linear dimension reduction techniques belonging to the class of PCA:

1. **Linear PCs**
   The simplest way is to define both g and h as the unity function, resulting in $\boldsymbol{W} = \boldsymbol{X}$ and $\boldsymbol{\kappa} = \boldsymbol{X}'\boldsymbol{X}$. Due to the linear link between the PCs and the data, PCA is straightforward to implement and yields consistent estimators for the latent factors if K and T go to infinity (Bai & Ng, 2008; Stock & Watson, 2002b). Even if there is some time-variation in the factor loadings (and K is large), Stock and Watson (2002a) show that principal components asymptotically (i.e., $T \to \infty$) remain a consistent estimator for the factors. They also show that the resulting forecast is efficient.[1]
2. **Quadratic and squared PCs**
   The literature suggests several ways to overcome the linearity restriction of PCs. Bai and Ng (2008), for example, apply a quadratic link function between the latent factors and the regressors, yielding a more flexible factor structure. While squared PC considers just squaring the elements of $\boldsymbol{X}$ resulting in

   $$\boldsymbol{W} = \boldsymbol{X}^2 \quad \text{and} \quad \boldsymbol{\kappa} = (\boldsymbol{X}^2)'(\boldsymbol{X}^2),$$

   with $\boldsymbol{X}^2 = (\boldsymbol{X} \odot \boldsymbol{X})$ and $\odot$ denoting element-wise multiplication, quadratic PC is defined as

   $$\boldsymbol{W} = (\boldsymbol{X}, \boldsymbol{X}^2) \quad \text{and} \quad \boldsymbol{\kappa} = \boldsymbol{W}'\boldsymbol{W}.$$

   Both variants focus on the second moments of the covariate matrix and allow for a non-linear relationship between the principal components and the predictors. Bai and Ng (2008) show that quadratic variables can have substantial predictive power as they provide additional information on the underlying time series. Intuitively speaking, given that we transform our data to stationarity in the empirical work, this transformation strongly overweights situations characterized by sharp movements in the columns of $\boldsymbol{X}$ (such as during a recession). By contrast, periods characterized by a slight variation in our macroeconomic panel are transformed to mildly fluctuate around zero (and thus carry little predictive content for inflation). In our empirical model, our regressions always feature lagged inflation and thus, this transformation effectively implies that the model is close to an autoregressive model in tranquil periods, whereas more information is introduced in crisis periods.
3. **Kernel PCs**
   Another approach for non-linear PCs is kernel principal component analysis (KPCA). KPCA dates back to Schölkopf et al. (1998), who proposed using integral operator kernel functions to compute PCs in a non-linear manner. This amounts to implicitly applying a non-linear transformation of the data through a kernel function and then applying PCA on this transformed dataset. Such an approach has been used for forecasting in Giovannelli (2012) and Exterkate et al. (2016).
   We allow for non-linearities in the kernel function between the data and the factors by defining h to be a Gaussian or a polynomial kernel $\boldsymbol{\kappa}$ (which is of dimension $K \times K$) with the (i, j)th element given by

   $$\kappa_{ij} = \exp\left(-\frac{\|\boldsymbol{x}_{\bullet i} - \boldsymbol{x}_{\bullet j}\|^2}{2c_1^2}\right)$$

   for a Gaussian kernel and

   $$\kappa_{ij} = \left(\frac{\boldsymbol{x}_{\bullet i}'\boldsymbol{x}_{\bullet j}}{c_0^2} + 1\right)^2$$

   for a polynomial kernel.
   Here, $\boldsymbol{W} = \boldsymbol{X}$ (i.e., g is the unity function), $\boldsymbol{x}_{\bullet i}$ and $\boldsymbol{x}_{\bullet j}$ $(i, j = 1, \ldots, K)$ denote two columns of $\boldsymbol{X}$, $\|\boldsymbol{x}_{\bullet i} - \boldsymbol{x}_{\bullet j}\|$ denotes the Euclidean distance between $\boldsymbol{x}_{\bullet i}$ and $\boldsymbol{x}_{\bullet j}$, while c_0 and c_1 are scaling parameters. As suggested by Exterkate et al. (2016) we set $c_0 = \sqrt{(K+2)/2}$ and $c_1 = \sqrt{c_K/\pi}$ with c_K being the 95th percentile of the χ^2 distribution with K degrees of freedom.

[1] Note that this result holds only asymptotically. However, with relatively small T and large K, forecast efficiency may be improved by capturing important non-linear features of the dataset.

2.2. Diffusion maps

Diffusion maps, originally proposed in Coifman et al. (2005) and Coifman and Lafon (2006), are another set of non-linear dimension reduction techniques that retain local interactions between data points in the presence of substantial non-linearities in the data.[2] The local interactions are preserved by introducing a random walk process.

The random walk captures the notion that moving between similar data points is more likely than moving to points that are less similar. We assume that the weight function that determines the strength of the relationship between $\boldsymbol{x}_{\bullet i}$ to $\boldsymbol{x}_{\bullet j}$ is given by

$$w(\boldsymbol{x}_{\bullet i}, \boldsymbol{x}_{\bullet j}) = \exp\left(\frac{\|\boldsymbol{x}_{\bullet i} - \boldsymbol{x}_{\bullet j}\|^2}{c_2}\right),$$

where $\|\boldsymbol{x}_{\bullet i} - \boldsymbol{x}_{\bullet j}\|$ denotes the Euclidean distance between $\boldsymbol{x}_{\bullet i}$ and $\boldsymbol{x}_{\bullet j}$ and c_2 is a tuning parameter set such that $w(\boldsymbol{x}_{\bullet i}, \boldsymbol{x}_{\bullet j})$ is close to zero except for $\boldsymbol{x}_{\bullet i} \approx \boldsymbol{x}_{\bullet j}$. Here, c_2 is determined by the median distance of the k-nearest neighbors of $\boldsymbol{x}_{\bullet i}$ as suggested by Zelnik-Manor and Perona (2004). The number of k is approximated using the algorithm suggested by Angerer et al. (2016).

The probability of moving from $\boldsymbol{x}_{\bullet i}$ to $\boldsymbol{x}_{\bullet j}$ is then simply obtained by normalizing:

$$p_{i\to j} = \text{Prob}(\boldsymbol{x}_{\bullet i} \to \boldsymbol{x}_{\bullet j}) = \frac{w(\boldsymbol{x}_{\bullet i}, \boldsymbol{x}_{\bullet j})}{\sum_j w(\boldsymbol{x}_{\bullet i}, \boldsymbol{x}_{\bullet j})}.$$

This probability tends to be small except for the situation where $\boldsymbol{x}_{\bullet i}$ and $\boldsymbol{x}_{\bullet j}$ are similar to each other. As a result, the probability that the random walk moves from $\boldsymbol{x}_{\bullet i}$ to $\boldsymbol{x}_{\bullet j}$ will be large if they are equal but small if both covariates differ strongly.

Let $\boldsymbol{P}$ denote a transition matrix of dimension $K \times K$ with (i, j)th element given by $p_{i\to j}$. The probability of moving from $\boldsymbol{x}_{\bullet i}$ to $\boldsymbol{x}_{\bullet j}$ in $n = 1, 2, \dots$ steps is then simply the matrix power of $\boldsymbol{P}^n$, with typical element denoted by $p^n_{i\to j}$. Using a biorthogonal spectral decomposition of $\boldsymbol{P}^n$ yields:

$$p^n_{i\to j} = \sum_{s\geq 0} \lambda_s^n \psi_s(\boldsymbol{x}_{\bullet i})\phi_s(\boldsymbol{x}_{\bullet j}),$$

with ψ_s and ϕ_s denoting left and right eigenvectors of $\boldsymbol{P}$, respectively. The corresponding eigenvalues are given by λ_s.

We then proceed by computing the so-called diffusion distance as follows:

$$\xi_n^2(\boldsymbol{x}_{\bullet i}, \boldsymbol{x}_{\bullet j}) = \sum_j \frac{(p^n_{i\to j} - p^n_{s\to j})^2}{p_0(\boldsymbol{x}_{\bullet j})},$$

with p_0 being a normalizing factor that measures the proportion the random walk spends at $\boldsymbol{x}_{\bullet j}$. This measure turns out to be robust with respect to noise and outliers. (Coifman & Lafon, 2006) show that

$$\xi_n^2(\boldsymbol{x}_{\bullet i}, \boldsymbol{x}_{\bullet j}) = \sum_{s=1}^{\infty} \lambda_s^{2n}(\psi_s(\boldsymbol{x}_{\bullet i}) - \psi_s(\boldsymbol{x}_{\bullet j}))^2.$$

This allows us to introduce the family of diffusion maps from $\mathbb{R}^K \to \mathbb{R}^q$ given by:

$$\boldsymbol{\Xi}_n(\boldsymbol{x}_{\bullet i}) = [\lambda_1^n \psi_1(\boldsymbol{x}_{\bullet i}), \dots, \lambda_q^n \psi_q(\boldsymbol{x}_{\bullet i})].$$

The distance matrix can then be approximated as:

$$\begin{aligned}\xi_n^2(\boldsymbol{x}_{\bullet i}, \boldsymbol{x}_{\bullet j}) &\approx \sum_{s=1}^{q} \lambda_s^{2n}(\psi_s(\boldsymbol{x}_{\bullet i}) - \psi_s(\boldsymbol{x}_{\bullet j}))^2 \\ &= \|\boldsymbol{\Xi}_n(\boldsymbol{x}_{\bullet i}) - \boldsymbol{\Xi}_n(\boldsymbol{x}_{\bullet j})\|^2.\end{aligned}$$

Intuitively, this equation states that we now approximate diffusion distances in $\mathbb{R}^K$ through the Euclidian distance between $\boldsymbol{\Xi}_n(\boldsymbol{x}_{\bullet i})$ and $\boldsymbol{\Xi}_n(\boldsymbol{x}_{\bullet j})$. This discussion implies that we have to choose n and q and we do this by setting $q = \{5, 15, 30\}$ according to our approach with either a small, moderate or large number of factors and $n = T$, the number of time periods. The algorithm in our application is implemented using the `R` packages `diffusionMap` and `destiny` (Angerer et al., 2016; Richards & Cannoodt, 2019).

2.3. Local linear embeddings

Locally linear embeddings (LLE) have been introduced by Roweis and Saul (2000). Intuitively, the LLE algorithm maps a high dimension input dataset $\boldsymbol{X}$ into a lower-dimensional space while preserving the neighborhood structure. This implies that points close to each other in the original space are also close in the transformed space.

The LLE algorithm is based on the assumption that each $\boldsymbol{x}_{\bullet i}$ is sampled from some underlying manifold. If this manifold is well defined, each $\boldsymbol{x}_{\bullet i}$ and its neighbors $\boldsymbol{x}_{\bullet j}$ are located close to a locally linear patch of this manifold. One consequence is that each $\boldsymbol{x}_{\bullet i}$ can be reconstructed from its neighbors $\boldsymbol{x}_{\bullet j}$ with $j \neq i$, conditional on suitably chosen linear coefficients. This reconstruction, however, will be corrupted by measurement errors. Roweis and Saul (2000) introduce a cost function to quantify these errors:

$$C(\boldsymbol{\Omega}) = \sum_i (\boldsymbol{x}_{\bullet i} - \sum_j \omega_{ij}\boldsymbol{x}_{\bullet j})^2,$$

with ω_{ij} denoting the (i, j)th element of a weight matrix $\boldsymbol{\Omega}$. This cost function is then minimized subject to the constraint that each $\boldsymbol{x}_{\bullet i}$ is reconstructed only from its neighbors. This implies that $\omega_{ij} = 0$ if $\boldsymbol{x}_{\bullet j}$ is not a neighbor of $\boldsymbol{x}_{\bullet i}$. The second constraint is that the matrix $\boldsymbol{\Omega}$ is row-stochastic, i.e., the rows sum to one. Conditional on these two restrictions, the cost function can be minimized by solving a least squares problem.

To make this algorithm operational, we need to define our notion of neighbors. In the following, we will use the k-nearest neighbors in terms of the Euclidean distance. We choose the number of neighbors by applying the algorithm proposed by Kayo (2006), which automatically determines the optimal number for k. The q latent factors in $\boldsymbol{Z}$, with typical ith column $\boldsymbol{z}_{\bullet i}$, are then obtained by minimizing:

$$\Phi(\boldsymbol{Z}) = \sum_i |\boldsymbol{z}_{\bullet i} - \sum_j \Omega_{ij}\boldsymbol{z}_{\bullet j}|^2,$$

[2] For an application to astronomical spectra, see Richards et al. (2009).

which implies a quadratic form in $\boldsymbol{z}_t$. Subject to suitable constraints, this problem can be easily solved by computing:

$$\boldsymbol{M} = (\boldsymbol{I}_T - \boldsymbol{\Omega})'(\boldsymbol{I}_T - \boldsymbol{\Omega}),$$

and finding the $q+1$ eigenvectors of $\boldsymbol{M}$ associated with the $q+1$ smallest eigenvalues. The bottom eigenvector is then discarded to arrive at q factors. For our application, we use the R package `lle` (Diedrich & Abel, 2012).

2.4. Isometric feature mapping

Isometric Feature Mapping (ISOMAP) is one of the earliest methods developed in the category of manifold learning algorithms. Introduced by Tenenbaum et al. (2000), the ISOMAP algorithm determines the geodesic distance on the manifold and uses multidimensional scaling to come up with a low number of factors describing the underlying dataset. Originally, ISOMAP was constructed for applications in visual perception and image recognition. In economics and finance, some recent papers highlight its usefulness (see, e.g., Lin et al., 2011; Orsenigo & Vercellis, 2013; Ribeiro et al., 2008; Zime, 2014).

The algorithm consists of three steps. In the first step, a dissimilarity index that measures the distance between data points is computed. These distances are then used to identify neighboring points on the manifold. In the second step, the algorithm estimates the geodesic distance between the data points as the shortest path distances. In the third step, metric scaling is performed by applying classical multidimensional scaling (MDS) to the matrix of distances. For the dissimilarity transformation, we determine the distance between point i and j by the Manhattan index $d_{ij} = \sum_k |x_{ki} - x_{kj}|$ and collect those points where i is one of the k-nearest neighbors of j in a dissimilarity matrix. For our empirical application, we again choose the number of neighbors by applying the algorithm proposed by Kayo (2006) and use the implementation in the R package `vegan` (Oksanen et al., 2019).

The described non-linear transformation of the dataset enables the identification of a non-linear structure hidden in a high-dimensional dataset and maps it to a lower dimension. Instead of pairwise Euclidean distances, ISOMAP uses the geodesic distances on the manifold and compresses information under consideration of the global structure.

2.5. Non-linear compression with deep learning

Deep learning algorithms are characterized by non-linearly converting input to output and representing the input itself in a transformed way. This is called representation learning because representations of the data are expressed in terms of other, simpler representations before mapping the data input to output values.

One tool that performs representation of itself and representation to output is the Autoencoder (AE). The first step is accomplished by the encoder function, which maps an input to an internal representation. The second part, which maps the encoded (transformed) data to the output, is called the decoder function. Their ability to extract factors (which explain a large fraction of the variability in the observed data) in a non-linear manner makes deep learners a powerful tool complementing the range of commonly used dimension reduction techniques (Goodfellow et al., 2016). Andreini et al. (2020), for example, embed a dynamic Autoencoder structure in a dynamic factor model and show that it yields a good now- and forecasting performance for US GDP. Their paper allows for additional flexibility by simultaneously estimating the non-linear latent factors and the parameters. In empirical finance, Heaton et al. (2017), Feng et al. (2018) and Kelly et al. (2019) find that the application of these methods is beneficial to predict asset returns.

Based on deep learning techniques, we propose obtaining hierarchical predictors $\boldsymbol{Z}$ by applying a number of $l \in \{1, \dots, L\}$ non-linear transformations to $\boldsymbol{X}$. These transformations are called hidden layers with L giving the depth of our architecture and f denoting an univariate activation function.[3] More specifically, in each layer, activation functions (non-linearly) transform the inputs (which are the outputs of the previous layer). A common choice, which we adopt, is the hyperbolic tangent (tanh) given by

$$f(\boldsymbol{X}) = \frac{\exp(\boldsymbol{X}) - \exp(-\boldsymbol{X})}{\exp(\boldsymbol{X}) + \exp(-\boldsymbol{X})}.$$

We apply this function element-wise to the entries of $\boldsymbol{X}$. Using tanh activation functions is justified by its strong empirical properties identified in recent studies such as Saxe et al. (2019) and Andreini et al. (2020).

The structure of our deep learning algorithm can be represented in form of a composition of univariate semi-affine functions given by

$$\hat{\boldsymbol{X}}^{(l)} = f\left(\hat{\boldsymbol{X}}^{(l-1)}\boldsymbol{W}^{(l-1)} + \boldsymbol{\iota}_T \otimes \boldsymbol{b}_{l-1}'\right)\boldsymbol{W}^{(l)} + \boldsymbol{\iota}_T \otimes \boldsymbol{b}_l', \quad \text{for } 1 \leq l \leq L,$$

and $\hat{\boldsymbol{X}}^{(0)} = \boldsymbol{X}$ for $l = 0$. Here, $\boldsymbol{W}^{(l)}$ denotes a weighting matrix of dimension $N_{l-1} \times N_l$ (with N_l being the number of neurons in layer l), $\boldsymbol{b}_l$ is a $N_l \times 1$ bias vector and $\boldsymbol{\iota}_T$ is a $T \times 1$ vector of ones.

The output of the network is then obtained by setting:

$$\boldsymbol{Z} = \hat{\boldsymbol{X}}^{(L)} = f\left(\hat{\boldsymbol{X}}^{(L-1)}\boldsymbol{W}^{(L-1)} + \boldsymbol{\iota}_T \otimes \boldsymbol{b}_{L-1}\right)\boldsymbol{W}^{(L)} + \boldsymbol{\iota}_T \otimes \boldsymbol{b}_L.$$

Notice that if we set $N_L = q (\ll K)$, we achieve dimension reduction, and the network's output is a (non-linearly) compressed version of the input dataset. In principle, what we have just described constitutes the encoding part of the Autoencoder. If we are interested in recovering the original dataset $\boldsymbol{X}$ we simply have to add additional layers characterized by increasing numbers of neurons until we reach $N_{L+j} = K$ for $j = 1, 2, \dots$.

The optimal sets of $\hat{\boldsymbol{W}} = (\hat{\boldsymbol{W}}^{(0)}, \dots, \hat{\boldsymbol{W}}^{(L)})$ and $\hat{\boldsymbol{b}} = (\hat{\boldsymbol{b}}_0, \dots, \hat{\boldsymbol{b}}_L)$ are obtained by computing a loss function, most commonly the mean squared error of the in-sample

[3] In principle, f can vary over the different layers.

fit. The complexity of the neural network is determined by choosing the number of hidden layers L and the number of neurons in each layer N_l. We perform our forecasting exercise with different sets of tuning parameters and choose one, three, five, and eight hidden layers with the number of neurons evenly being downsized to the desired number of factors.

For the loss function and the optimization algorithm, we stick to common choices in the literature and use the mean squared error loss function and the Adaptive Moment Estimation (ADAM). We repeat the optimization procedure in 100 epochs on at least 84 batches, which corresponds to the average duration of a business cycle in the US.[4] This implies that we train the algorithm in each epoch with a partition of the original data set of at least the length of one business cycle. The optimization procedure needs to be repeated in a reasonably high number of epochs to capture the dynamics of the different cycles present in the data. We find that the algorithm converges quickly, and setting the number of epochs to 100 is sufficient.

We employ the R interface to `keras` (Allaire & Chollet, 2019), a high-level neural networks API and widely used package for implementing deep learning models.

3. A TVP regression for forecasting inflation

In the following, we introduce the predictive regression that links our target variable, inflation in consumer prices, to $\boldsymbol{Z}$ and other observed factors. Following Stock and Watson (1999), inflation is specified such that:

$$y_{t+h} = \log\left(\frac{\text{CPI}_{t+h}}{\text{CPI}_t}\right) - \log\left(\frac{\text{CPI}_t}{\text{CPI}_{t-1}}\right), \tag{2}$$

with CPI_{t+h} denoting the consumer price index in period $t+h$.

In the empirical application we set $h \in \{1, 3\}$. y_{t+h} is then modeled using a dynamic regression model:

$$y_{t+h} = \boldsymbol{d}_t'\boldsymbol{\beta}_{t+h} + \epsilon_{t+h}, \quad \epsilon_{t+h} \sim \mathcal{N}(0, \sigma^2_{t+h}), \tag{3}$$

where $\boldsymbol{\beta}_{t+h}$ is a vector of TVPs associated with $M(= q + p)$ covariates denoted by $\boldsymbol{d}_t$ and σ^2_{t+h} is a time-varying error variance. $\boldsymbol{d}_t$ might include the latent factors extracted from the various methods discussed in the previous subsection, lags of inflation, an intercept term or other covariates that are not compressed.

Following much of the literature (Belmonte et al., 2014a; Chan, 2017; Huber et al., 2021; Kalli & Griffin, 2014; Kastner & Frühwirth-Schnatter, 2014; Stock & Watson, 2016; Taylor, 1982) we assume that the TVPs and the error variances evolve according to independent stochastic processes:

$$\begin{pmatrix} \boldsymbol{\beta}_{t+h} \\ \log\sigma^2_{t+h} \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \boldsymbol{\beta}_{t+h-1} \\ \mu_h + \rho_h \log\sigma^2_{t+h-1} \end{pmatrix}, \begin{pmatrix} \boldsymbol{V} & 0 \\ 0 & \vartheta^2_h \end{pmatrix}\right), \tag{4}$$

with μ_h denoting the conditional mean of the log-volatility, ρ_h its persistence parameter and ϑ^2_h the error variance of $\log\sigma^2_{t+h}$. The matrix $\boldsymbol{V}$ is an $M \times M$-dimensional variance–covariance matrix with $\boldsymbol{V} = \text{diag}(v^2_1, \ldots, v^2_M)$ and v^2_j being the process innovation variance that determines the amount of time-variation in $\boldsymbol{\beta}_{t+h}$. This setup implies that the TVPs are assumed to follow a random walk process while the log-volatilities evolve according to an AR(1) process.

The model described by Eqs. (3) and (4) is a flexible state space model that encompasses a wide range of models commonly used for forecasting inflation. For instance, if we set $\boldsymbol{V} = \boldsymbol{0}_M$ and $\vartheta^2 = 0$, we obtain a constant parameter model with homoscedastic errors. If $\boldsymbol{V}$ is instead a full $M \times M$ matrix, but of reduced rank, we obtain the model proposed in Chan et al. (2020). If $\boldsymbol{d}_t$ includes the lags of inflation and (lagged) PCs, we obtain a model closely related to the one used in Stock and Watson (2002b). If we set $d_t = 1$ and allow for TVPs, we obtain a specification similar to the unobserved components stochastic volatility model successfully adopted in Stock and Watson (1999). Many other models can be identified by appropriately choosing $\boldsymbol{d}_t$, $\boldsymbol{V}$, and ϑ^2. This flexibility, however, calls for model selection. We select appropriate submodels by using Bayesian methods for estimation and forecasting. These techniques are further discussed in Section B of the Online Appendix and allow for data-based shrinkage towards simpler nested alternatives.

4. Forecasting US inflation

4.1. Data overview, design of the forecasting exercise and competitors

In our empirical application, we consider the popular FRED-MD database. This dataset is publicly accessible and available in real-time. The monthly data vintages ensure that we only use information that would have been available at the time a given forecast is being produced. A detailed description of the databases can be found in McCracken and Ng (2016). To achieve approximate stationarity, we transform the dataset as outlined in Section C of the Online Appendix. Furthermore, each time series is standardized to have a sample mean zero and unit sample variance before using the non-linear dimension reduction techniques.

Our US dataset includes 105 monthly variables that span the period from 1963:01 to 2021:01. The forecasting design relies on a rolling window, as justified in Clark (2011), that initially ranges from 1980:01 to 1999:12. For each month of the hold-out sample, which starts in 2000:01 and ends in 2019:12, we compute the h-month-ahead predictive distribution for each model (for $h \in \{1, 3\}$), keeping the length of the estimation sample fixed at 240 observations (i.e., a rolling window of 20 years).[5] For these periods, we contrast each forecast with

[4] The average duration of a business cycle was determined using data provided by The National Bureau of Economic Research (NBER) on business cycle expansions and recessions.

[5] In addition to our baseline sample ending in 2019:12, we present the results of our forecasting exercise, including observations covering the COVID-19 pandemic (2020:01 to 2020:08) in Section 4.5. Since the pandemic caused severe outliers in our dataset, including those periods helps to test the forecasting performance of our models during turbulent times.

the realization of inflation in the vintage one-quarter-ahead, following the evaluation approach of Chan (2017). In this way, we make sure that realized inflation is not subject to revisions anymore as most data revisions take place in the first quarter while afterward, the vintages remain relatively unchanged (see, e.g., Croushore, 2011; Pfarrhofer, 2020).

One key limitation is that all methods are specified conditionally on $\boldsymbol{d}_t$ and thus implicitly on the specific function f used to move from $\boldsymbol{X}$ to $\boldsymbol{Z}$. Another key objective of this paper is to control uncertainty with respect to f by using dynamic model averaging techniques. We use the first 24 observations of our hold-out sample to obtain an initial combined predictive density for the period 2002:01. The remaining periods (i.e., ranging from 2002:01 to 2019:12) then constitute our evaluation sample, and the respective predictions are again contrasted to the one-quarter-ahead vintage of inflation.

In terms of competing models we can classify the specifications along two dimensions:

1. **How $\boldsymbol{d}_t$ is constructed.** First, let $\boldsymbol{s}_t$ denote a K_0-dimensional vector of covariates except for y_t. $\boldsymbol{x}_t = (\boldsymbol{s}_t', \ldots, \boldsymbol{s}_{t-p+1}')'$ is then composed of p lags of s_t with $K = pK_0$. In our empirical work we set $p = 12$ and include all variables in the dataset (except for the transformed CPI series, i.e., $K_0 = 104$). We then use the different dimension reduction techniques outlined in Section 2 to estimate $\boldsymbol{z}_t$. Moreover, we include p lags of y_t as additional observed factors to $\boldsymbol{d}_t$. This serves to investigate how different dimension reduction techniques perform when interest centers on predicting inflation. We also consider simple AR(12) models as well as a small- and a large-scale AR specification augmented with (observed) exogenous covariates (henceforth labeled ARX) as additional competitors. For the small-scale variants we include five exogenous regressors, while for the large-scale ARX model we use 20 additional covariates. Since the macroeconomic forecasting literature is quite inconclusive about variable inclusion in such predictive ARX models for inflation (see, e.g., De Mol et al., 2008; Hauzenberger et al., 2019; Koop & Korobilis, 2012; Stock & Watson, 2008), we use a semi-automatic approach that handles this issue rather agnostically. We discuss this in more detail in Section 4.2.
2. **The relationship between $\boldsymbol{d}_t$ and y_{t+h}.** The second dimension along which our models differ is the specific relationship described by Eq. (3). To investigate whether non-linear dimension reduction techniques are sufficient to control for unknown forms of non-linearities, we benchmark all our models that feature TVPs with their respective constant parameter counterparts. To perform model selection, we consider two priors. The first one is the horseshoe (HS, Carvalho et al., 2010) prior and the second one is an adaptive Minnesota (MIN, see Carriero et al., 2015; Chan, 2021; Giannone et al., 2015) prior (for further details see Section B of the Online Appendix).

4.2. Properties of the factors

In this subsection, we analyze bivariate correlations between the factors obtained from using different dimension reduction techniques and the variables in our dataset, as well as inflation. These correlations provide information on the specific factor dynamics and (with caution) on how to interpret the factors in $\boldsymbol{Z}$ from a structural perspective.[6] The recent literature (Crawford et al., 2019, 2018; Joseph, 2019) advocates using linear approximations or Shapley values to improve the interpretability of these highly non-linear models. In this paper, we opt for a simple correlation-based approach given the large amount of competing dimension reduction techniques and the fact that for some of these, the different techniques work better than for other methods.

Fig. 1 is a heatmap of the correlations with rows denoting the different covariates in $\boldsymbol{X}$ and columns representing the different dimension reduction techniques. These correlations are averages across the factors (in case that $q > 1$) and, since we include several lags of the input dataset, are also averaged across the lags.

The figure suggests for most dimension reduction techniques that the factors are correlated with housing quantities (PERMIT and HOUST alongside their subcomponents) as well as interest rate spreads. Some variables that measure real activity (such as industrial production and several of its components) also display comparatively large correlations with the factors. In some cases, these correlations are positive, whereas in other cases, correlations are negative. In both instances, however, the absolute magnitudes are similar. The three exceptions from this rather general pattern are diffusion maps as well as PCA quadratic and squared. In this case, the corresponding columns indicate lower correlations.

Averaging over the factors, as done in Fig. 1, potentially masks important features of individual factors. Next, we ask whether there are relevant differences by analyzing the correlations between each $\boldsymbol{z}_j$ ($j = 1, \ldots, q$) and each column of $\boldsymbol{X}$. For brevity, we focus on a specific model that performs extraordinarily well in terms of density forecasts: the Autoencoder with a single hidden layer and 30 factors. Fig. 2 shows, for each factor, the five variables that display the largest absolute correlation. The variables in the rows are a union over the sets of top-five variables for each factor. This figure shows that several factors display quite similar correlation patterns. For all of them, housing quantities are positively or negatively correlated (with similar magnitudes). Apart from that, and consistent with the findings discussed above, we observe that financial market variables (such as interest rate spreads) show up frequently for several factors. Only very few factors depart from this overall pattern. In the case of factors 9, 22, 23, and 24, we find low correlations with housing and much stronger correlations with financial markets. Factor 9 closely tracks the credit (BAAFFM) and term spreads (e.g., T10YFFM).

[6] The estimates of the factor are considerably more difficult to interpret. Nevertheless, to provide some intuition on how the factors for the best performing specifications evolve, see Figure A.1 in the Online Appendix.

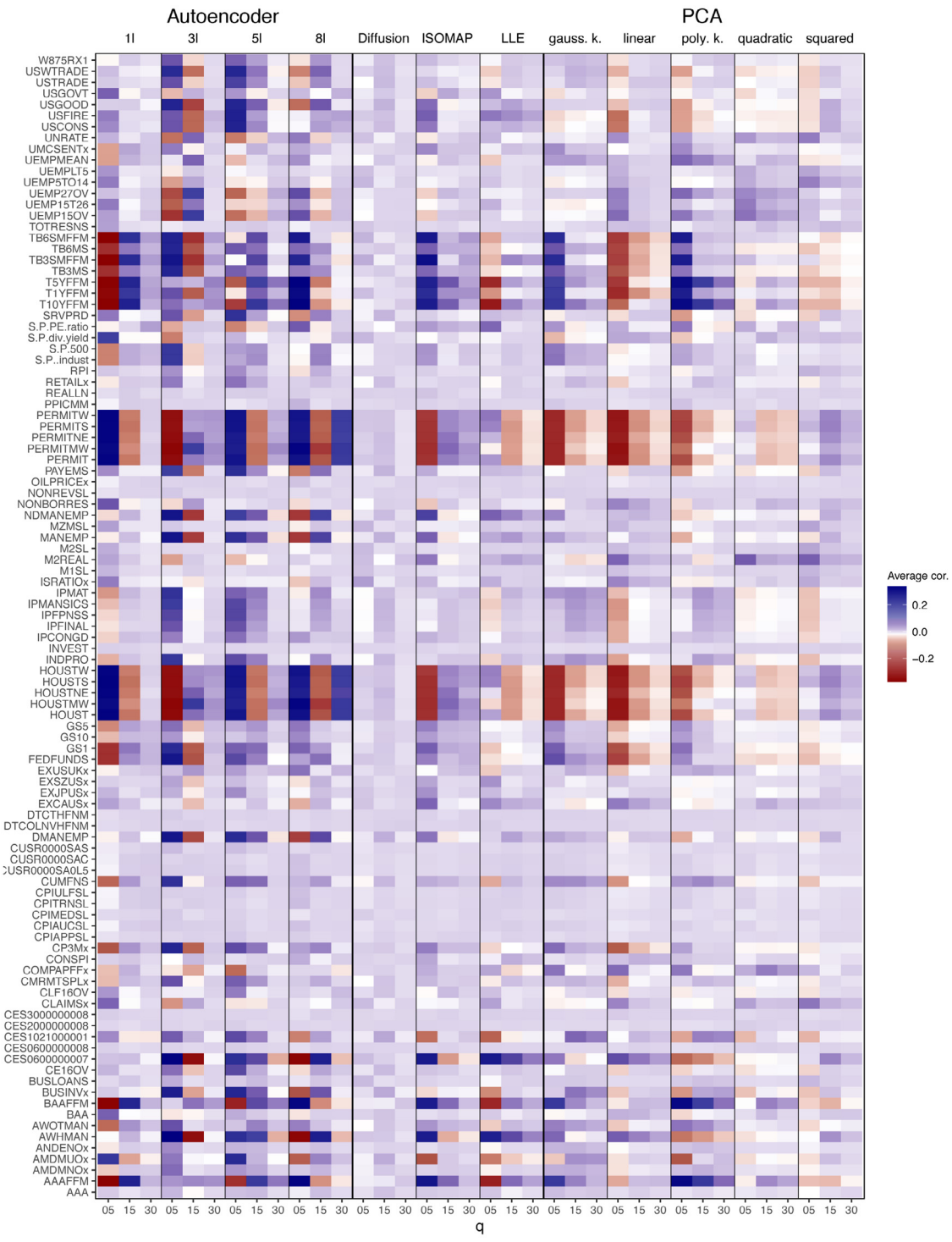

Fig. 1. Correlations between the variables in the input dataset and the factors obtained from the different dimension reduction techniques.

These two heatmaps provide a rough overview of what variables drive the factors. Next, we ask whether we can construct models based on including variables that display the strongest correlations with the factors. This approach can be interpreted as a simple selection device that takes non-linearities in the input dataset implicitly into account. Since the heatmap is based on full-sample results and we are interested in using these small-scale

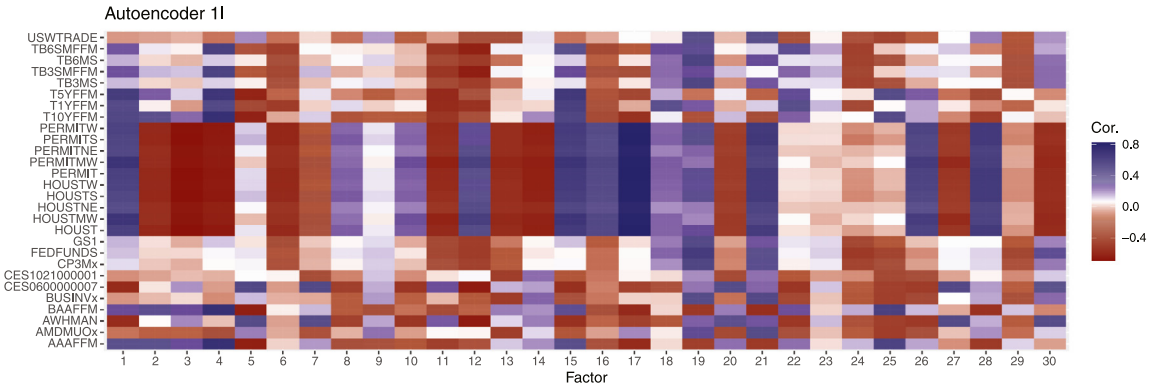

Fig. 2. Correlations between the top-five variables in the input dataset and the factors obtained from the Autoencoder with one layer.

models for out-of-sample forecasting, we compute the correlation for each point in our hold-out period. In summary, the variables that frequently show up across hold-out periods and dimension reduction techniques are:

- **Real activity and housing:** Variables on industrial production (INDPRO, IPMANSICS), capacity utilization (CUMFNS) and private housing starts (HOUST) and permits (PERMIT),
- **Labor market:** Variables on (un-)employment (MANEMP, USGOOD) and average hours worked (CES0600000007, AWHMAN),
- **Prices:** Sub-indicators of consumer prices (CUSR0000SA0L5),
- **Interest rates and other stock market variables:** Spreads (to the Fed funds rate) of treasuries (TB3SMFFM, TB6SMFFM, T1YFFM, T10YFFM) and of corporate bonds (AAAFFM, BAAFFM, COMPAPFFx),
- **Money stocks and reserves:** Non-borrowed reserves (NONBORRES) and adjusted monetary base (AMBSL).

These variables are also the ones that display high correlations to the factors in Fig. 1 and are included in the large-scale ARX model. Here, it is worth stressing that there seems to be appreciable heterogeneity with respect to dimension reduction methods. Most of them generate factors that are highly correlated with real activity and housing measures as well as interest rates and other stock market variables. Interestingly, when we focus on the second group, we observe that the factors arising from using PCA squared (and to a somewhat lesser extent PCA quadratic) are heavily related to labor market measures. Average correlations with prices (i.e., CUSR0000SA0L5) are small for most techniques (PCA quadratic yields the largest correlations of around $0.3 - 0.4$). Some methods also yield factors strongly correlated to money stocks and reserves (e.g., diffusion maps). Table C.3 of the Online Appendix provides a much more detailed picture of the precise variables used to build the small-scale models.

Next, we ask whether the factors are correlated to inflation. Table 1 shows the correlation with inflation averaged across the number of factors for each dimension reduction technique, as well as the minimum and maximum value (across these factors) in parentheses. To assess whether these correlations differ over time, we divide our sample into expansionary and recessionary periods.[7] Since the COVID-19 pandemic marks an extraordinary period in our sample, we also compute the correlations for 2020 only and include it at the bottom of Table 1.

For the full sample as well as during expansions, we find that the factors obtained from using the linear variants of PCA display comparatively higher correlations relative to the other dimension reduction techniques (with some of the factors featuring a correlation of close to 0.2). In recessions and the pandemic, these correlations increase substantially to reach average correlations close to 0.3 (with the factor displaying the maximum correlation being strongly related to inflation, with values of around 0.6). The non-linear dimension reduction techniques yield strong correlations during turbulent times (i.e., recessions and the pandemic). This is not surprising since these methods tend to work well if there are strong deviations from linearity (which mainly occurs in recessions). Such a feature can be easily demonstrated by considering PCA squared. The factors will be centered around zero in normal times and typically display little variation. But in recessions, the link function implies that larger changes will dominate the shape of the factors and imply pronounced movements that could help predict turning points in inflation.

4.3. Density and point forecast performance

We now consider point and density forecasting performance of the different models and dimension reduction techniques. The forecast performance is evaluated through log predictive likelihoods (LPLs) for density forecasts, and root mean squared errors (RMSEs) for point forecasts. Superior models are those with high scores in terms of LPL and low values in terms of RMSE. We benchmark all models relative to the autoregressive (AR) model with constant parameters and the Minnesota prior. The first entry in the tables gives the actual value of the LPL (cumulated over the hold-out sample) with actual RMSEs in parentheses (averaged over the hold-out sample) for our benchmark model. The remaining entries

[7] Recessions are defined by using the business cycle classification of the National Bureau of Economic Research (NBER).

Table 1
Average correlation with inflation.

Business Cycle	No. of factors	PCA linear	PCA quadratic	PCA squared	PCA gauss. kernel	PCA poly. kernel	ISOMAP	Diffusion Maps	LLE	Autoencoder 1l
Full Sample	q = 05	0.017	**0.114**	0.113	0.013	0.016	0.007	0.061	0.009	0.008
	q = 05	(0.003,0.043)	(0.006,0.251)	(0.006,0.250)	(0.004,0.024)	(0.003,0.035)	(0.002,0.010)	(0.002,0.163)	(0.003,0.017)	(0.002,0.018)
	q = 15	**0.086**	0.069	0.068	0.030	0.069	0.034	0.052	0.013	0.035
	q = 15	(0.003,0.198)	(0.003,0.251)	(0.005,0.250)	(0.004,0.133)	(0.003,0.206)	(0.003,0.095)	(0.001,0.163)	(0.001,0.035)	(0.004,0.090)
	q = 30	**0.108**	0.049	0.050	0.052	0.106	0.043	0.063	0.036	0.036
	q = 30	(0.002,0.259)	(0.003,0.251)	(0.005,0.250)	(0.004,0.274)	(0.003,0.292)	(0.001,0.143)	(0.001,0.185)	(0.001,0.168)	(0.001,0.163)
Expansion	q = 05	0.027	**0.069**	0.055	0.033	0.030	0.021	0.045	0.010	0.023
	q = 05	(0.016,0.049)	(0.003,0.151)	(0.003,0.110)	(0.013,0.059)	(0.019,0.050)	(0.009,0.050)	(0.022,0.098)	(0.003,0.02)	(0.006,0.043)
	q = 15	**0.073**	0.057	0.055	0.030	0.061	0.035	0.049	0.020	0.039
	q = 15	(0.001,0.192)	(0.003,0.151)	(0.003,0.110)	(0.003,0.087)	(0.009,0.194)	(0.003,0.118)	(0.005,0.098)	(0.003,0.068)	(0.003,0.115)
	q = 30	**0.103**	0.047	0.045	0.049	0.098	0.040	0.053	0.043	0.041
	q = 30	(0.001,0.290)	(0.001,0.151)	(0.001,0.110)	(0.001,0.276)	(0.009,0.285)	(0.001,0.134)	(0.005,0.116)	(0.003,0.209)	(0.001,0.196)
Recession	q = 05	0.098	**0.250**	0.248	0.121	0.116	0.042	0.180	0.065	0.071
	q = 05	(0.040,0.156)	(0.123,0.442)	(0.123,0.442)	(0.049,0.149)	(0.051,0.163)	(0.014,0.091)	(0.122,0.269)	(0.007,0.114)	(0.012,0.190)
	q = 15	**0.152**	0.137	0.136	0.097	0.143	0.090	0.133	0.067	0.068
	q = 15	(0.004,0.286)	(0.012,0.442)	(0.016,0.442)	(0.003,0.321)	(0.022,0.326)	(0.015,0.205)	(0.019,0.269)	(0.003,0.172)	(0.010,0.188)
	q = 30	0.152	0.110	0.114	0.118	**0.158**	0.104	0.132	0.081	0.077
	q = 30	(0.004,0.297)	(0.009,0.442)	(0.011,0.442)	(0.003,0.321)	(0.022,0.424)	(0.005,0.366)	(0.003,0.306)	(0.003,0.180)	(0.003,0.297)
Pandemic	q = 05	0.320	0.285	0.287	0.124	0.231	0.199	**0.367**	0.303	0.087
	q = 05	(0.059,0.631)	(0.188,0.424)	(0.216,0.423)	(0.020,0.210)	(0.017,0.642)	(0.101,0.303)	(0.127,0.610)	(0.087,0.506)	(0.046,0.131)
	q = 15	0.292	**0.322**	0.314	0.292	0.260	0.173	0.308	0.228	0.126
	q = 15	(0.059,0.631)	(0.015,0.641)	(0.012,0.642)	(0.020,0.734)	(0.008,0.642)	(0.022,0.559)	(0.053,0.610)	(0.088,0.402)	(0.017,0.365)
	q = 30	0.269	0.348	0.320	0.260	0.262	0.261	0.313	**0.358**	0.160
	q = 30	(0.029,0.631)	(0.015,0.641)	(0.010,0.642)	(0.020,0.796)	(0.008,0.646)	(0.001,0.764)	(0.001,0.886)	(0.057,0.821)	(0.007,0.475)

Note: The values are averaged across the number of factors stated in the second column with minimum and maximum values in parentheses. All correlation values are absolute values. The periods are divided into business cycle phases according to the National Bureau of Economic Research (NBER, https://www.nber.org/research/business-cycle-dating).

are differences in LPLs with relative RMSEs in parentheses. We mark statistically significant results according to the Diebold and Mariano (1995) test at the one, five, and ten percent significance levels with one, two, and three asterisks, respectively.

Starting with the one-month-ahead horizon, Table 2 depicts the inflation forecasting results. This table suggests that using dimension reduction techniques (both linear and non-linear) improves predictions substantially in terms of density forecasts. These improvements arise not only relative to the AR benchmark but also related to the large AR models with additional exogenous regressors. For some models, these improvements are sizable, irrespective of the regression specification (i.e., whether we use a constant parameter or a TVP model). Especially the Autoencoder with one and five layers sharply improves upon the benchmark (and all the remaining competitors) by large margins. Moreover, it yields statistically significant improvements at the one percent level. A similar story emerges when we focus on point forecasts. Non-linear dimension reductions help slightly. Relative RMSEs are smaller but close to one for most models. Again, the Autoencoder works well and yields RMSEs that are, across regression specifications, almost 20 percent lower than the ones from the benchmark. It is worth emphasizing that PCA squared also yields highly competitive point forecasts that are statistically significant according to the Diebold and Mariano (1995) test.

When comparing model performance across regression specifications and focusing on the Minnesota-type priors, we find that constant parameter models work quite well if non-linear dimension reduction techniques such as the Autoencoder are adopted. With a single exception (diffusion maps), introducing TVPs does not pay off and yields density forecasts that are slightly more imprecise than those obtained from their time-invariant counterparts. If we use a horseshoe prior, this result somewhat reverses (with the caveat that the models coupled with the horseshoe sometimes yield weaker inflation forecasts than the benchmark). Here, we observe that introducing TVPs often improves log predictive likelihoods relative to the constant parameter model with the same prior.

Summing up this discussion, the Autoencoder yields favorable point and density forecasts, irrespective of the prior and regression specification chosen. However, this strong performance of the Autoencoder depends on the number of layers and the number of factors. The literature (see, e.g., Heaton, 2008; Huang, 2003) suggests that the number of hidden layers should increase with the complexity of the dataset. Our results, however, indicate the opposite. For a typical US macroeconomic dataset, the forecast performance of the Autoencoder seems to be strongest when a single hidden layer coupled with a large number of factors is used.

Next, we inspect the longer forecast horizon in greater detail. Table 3 depicts the forecast performance of all competitors for the one-quarter-ahead horizon. The table indicates that several non-linear dimension reduction techniques (most notably the Autoencoder, PCA quadratic, and PCA squared) outperform the autoregressive benchmark as well as models based on linear PCs. The improvements relative to linear PCs are sizable and statistically significant (especially for squared and quadratic PCs). For this horizon, the large ARX models also exhibit excellent forecasting properties.

Zooming into the different approaches to dimension reduction reveals that PCA quadratic with TVPs yields the highest LPLs. PCA quadratic and squared stand out among the different dimension reduction techniques and improve appreciably against all other dimension reduction techniques. The Autoencoder also provides highly competitive density forecasts.

When we consider point forecasts, a similar picture arises. Here we find that the models that do well in terms of LPLs also yield precise point forecasts. However, the best performing specification is PCA squared with five factors, constant parameters, and a horseshoe prior. This model improves upon the AR benchmark by around 24 percent. Notice, however, that the same model but with TVPs also yields 24 percent more precise forecasts than

Table 2
One-month-ahead forecast performance.

Specification	const. (MIN)	const. (HS)	TVP (MIN)	TVP (HS)
AR	−329.28	0.23	−0.99	−1.04
	(1.24)	(0.98)	(0.98)	(0.98)
Large ARX	2.46	−9.80***		−8.95***
	(0.97)	(1.06***)		(1.04***)
Autoencoder 1l (q = 05)	2.19	−2.06	1.12	−1.43
	(0.98)	(0.99)	(0.97)	(0.99)
Autoencoder 1l (q = 15)	16.26***	12.71***	21.41***	13.69***
	(0.90***)	(0.91***)	(0.87***)	(0.90***)
Autoencoder 1l (q = 30)	**38.19*** **	**29.92*** **	**35.48*** **	**31.39*** **
	(**0.81*** **)	(**0.84*** **)	(**0.80*** **)	(**0.83*** **)
Autoencoder 3l (q = 05)	3.55	−1.59	1.90	−2.35
	(0.97)	(0.99)	(0.97)	(0.99)
Autoencoder 3l (q = 15)	11.52***	8.90**	13.19**	8.35**
	(0.94**)	(0.95*)	(0.92**)	(0.95**)
Autoencoder 3l (q = 30)	23.63***	16.44***	18.62*	12.81*
	(0.86***)	(0.88***)	(0.83***)	(0.87***)
Autoencoder 5l (q = 05)	2.50	−2.07	1.55	−2.74
	(0.97)	(0.99)	(0.97)	(0.99)
Autoencoder 5l (q = 15)	10.05*	4.45	11.65*	5.62
	(0.94**)	(0.95)	(0.93**)	(0.94*)
Autoencoder 5l (q = 30)	26.91***	22.80***	26.72***	23.12***
	(0.85***)	(0.86***)	(0.84***)	(0.86***)
Autoencoder 8l (q = 05)	1.79	−2.58	3.27	−3.05
	(0.97)	(0.99)	(0.97)	(0.99)
Autoencoder 8l (q = 15)	2.86	−4.34	1.49	−2.52
	(0.98)	(1.00)	(0.97)	(1.00)
Autoencoder 8l (q = 30)	5.64	−0.97	7.21*	0.13
	(0.97)	(0.99)	(0.95)	(0.98)
Diffusion Maps (q = 05)	2.12	−3.64	2.97	−2.40
	(0.98)	(1.00)	(0.98)	(1.00)
Diffusion Maps (q = 15)	2.10	−8.17**	2.13	−7.54**
	(0.98)	(1.03**)	(0.99)	(1.03**)
Diffusion Maps (q = 30)	2.55	−9.13**	3.81	−6.79*
	(0.98)	(1.03**)	(0.97)	(1.02**)
ISOMAP (q = 05)	2.55	−3.36	2.28	−1.24
	(0.98)	(0.99)	(0.97)	(0.99)
ISOMAP (q = 15)	3.01	−5.33*	1.31	−5.46*
	(0.97)	(1.01)	(0.98)	(1.01*)
ISOMAP (q = 30)	1.50	−6.04**	2.36	−4.53
	(0.97)	(1.01**)	(0.97)	(1.01)
LLE (q = 05)	1.39	−2.23	1.93	−1.84
	(0.98)	(0.99)	(0.98)	(1.00)
LLE (q = 15)	2.68	−5.87**	0.83	−5.18*
	(0.97)	(1.02*)	(0.97)	(1.01)
LLE (q = 30)	1.86	−11.38**	0.45	−11.49**
	(0.98)	(1.02*)	(0.98)	(1.01*)
PCA gauss. kernel (q = 05)	2.44	−1.37	1.50	0.07
	(0.98)	(0.99)	(0.97)	(0.98)
PCA gauss. kernel (q = 15)	2.97	−1.30	1.85	−3.88
	(0.98)	(0.99)	(0.98)	(0.99)
PCA gauss. kernel (q = 30)	2.68	−2.94	3.10	−1.36
	(0.97)	(1.00)	(0.97)	(0.99)
PCA linear (q = 05)	1.96	−3.74	1.27	−2.46
	(0.98)	(1.00)	(0.98)	(0.99)
PCA linear (q = 15)	3.78	−4.89	3.35	−3.76
	(0.97)	(1.01)	(0.97)	(1.00)
PCA linear (q = 30)	3.25	−7.51**	2.10	−5.38
	(0.97)	(1.02**)	(0.98)	(1.02**)
PCA poly. kernel (q = 05)	3.74	−1.06	2.50	−0.76
	(0.98)	(0.99)	(0.98)	(0.99)
PCA poly. kernel (q = 15)	1.01	−1.90	2.37	−2.17
	(0.98)	(0.99)	(0.98)	(0.99)
PCA poly. kernel (q = 30)	4.28	−1.78	2.51	−2.43
	(0.97)	(0.99)	(0.97)	(0.99)

(*continued on next page*)

Table 2 (*continued*).

PCA quadratic (q = 05)	10.56**	7.15	9.72*	8.32
	(0.91*)	(0.93)	(0.90*)	(0.93)
PCA quadratic (q = 15)	4.63	0.01	6.79	1.34
	(1.00)	(0.99)	(0.97)	(0.98)
PCA quadratic (q = 30)	3.15	−7.23	5.11	−4.72
	(0.97)	(1.04**)	(0.96)	(1.03*)
PCA squared (q = 05)	9.78*	8.44	11.30*	7.10
	(0.90*)	(0.92)	(0.89*)	(0.92)
PCA squared (q = 15)	5.77	0.70	7.36	2.47
	(0.97)	(0.98)	(0.93)	(0.98)
PCA squared (q = 30)	4.33	−1.49	6.79*	−3.58
	(0.97)	(1.01)	(0.95)	(1.02)

Note: The table shows log predictive likelihoods (LPLs) with root mean squared errors (RMSEs) in parentheses below. The first (red shaded) entry gives the actual LPL and RMSE scores of our benchmark (an autoregressive (AR) model with constant parameters and a Minnesota prior). Asterisks indicate statistical significance for each model relative to the benchmark at the 1% (***), 5% (**) and 10% (*) significance levels. Since the large ARX model with time-varying parameters would feature 273 period-specific coefficients and is computationally intractable, we assume that the TVPs feature a factor structure (with three factors) to reduce the dimension of the state space (see Section B of the Online Appendix and Chan et al., 2020).

the ones of the benchmark. This suggests that the horseshoe shrinks the TVPs close to zero, and the corresponding point estimators are almost identical. Since the LPLs differ, the remaining time variation in the coefficients mainly impacts the LPLs through the predictive variance.

Table 4 provides a summary of the best performing models for the one-month and the one-quarter-ahead forecasts, respectively. Moreover, we also assess how a model that includes the five variables displaying the highest average correlations to the factors performs (see discussion in Section 4.2). These models are labeled small ARX in the table. The results suggest that for one-month-ahead forecasts, replacing the latent factors arising from the Autoencoder with observed variables that display a high correlation to the factors does not pay off in terms of point and density forecasts. Across all regression specifications, the RMSEs are higher and the LPLs lower. However, when we consider one-quarter-ahead forecasts, this strategy seems to work much better. In this case, smaller models with covariates selected based on their correlations to the factors obtained using PCA squared and PCA quadratic yield density predictions close to those obtained from exploiting all available data. Notice, however, that for point forecasts, the performance of the small models is inferior.

Before proceeding to the next subsection, we briefly discuss two important issues. First, it is worth stressing that the factors used in this forecasting exercise are extracted from the full set of variables in $\boldsymbol{X}$. In Table A.1 and Table A.2 of the Online Appendix we divide the dataset into slow- and fast-moving variables (Bernanke et al., 2005) and extract the latent factors from these partitioned datasets exclusively. The main results based on extracting the factors from the full dataset remain in place. The Autoencoder performs particularly well for one-month-ahead forecasts, whereas PCA squared and quadratic yield accurate forecast densities for one-quarter-ahead predictions.

Second, for one of our best-performing models (the Autoencoder with one hidden layer), forecasting performance changes sharply when the number of factors is changed. This raises questions about the relationship between the number of factors and the forecast performance. In Figure A.2 in the Online Appendix, we show two graphs that discuss how point and density forecasting performance change with the number of factors. In this exercise, we find that the largest jumps in predictive accuracy are found when increasing the number of factors from 17 to 24 and 29 to 30 in terms of LPLs, and from 18 to 26 and 29 to 30 in terms of RMSE.

4.4. Assessing model calibration using probability integral transforms

The results based on RMSEs and LPLs provide information on relative forecasting performance. In the next step, we ask whether the different methods and models that we propose yield predictive distributions that are better calibrated. To this end, we consider the normalized forecast errors obtained through the probability integral transform (PIT). If a model is correctly specified, the PITs are iid uniformly distributed, and the respective standardized forecast errors should be iid normally distributed. Departures from the standard Gaussian distribution allow us to inspect the dimensions for which the model is poorly calibrated. For instance, if the variance of the normalized forecast error is too small (i.e., below one), this is evidence that the predictive distribution is too wide (i.e., too many predictions are in the tails). At the same time, values greater than one indicate that the variance is too tight (i.e., the tails are not adequately represented).

Table 5 shows the results for the one-month-ahead normalized forecast errors.[8] In principle, we observe that the mean across methods is close to zero (with a few exceptions such as PCA quadratic for $q = 15$). Nevertheless, these differences are never statistically significantly different from zero. Considering the variances shows that most models yield forecast distributions that seem slightly too narrow (with variances exceeding one).

[8] The results for one-quarter-ahead forecasts are provided in Section A of the Online Appendix.

Table 3
One-quarter-ahead forecast performance.

Specification	const. (MIN)	const. (HS)	TVP (MIN)	TVP (HS)
AR	−322.66	9.79	20.02	18.53
	(1.27)	(0.94*)	(0.89*)	(0.89**)
Large ARX	19.68	14.44		11.03
	(0.89*)	(0.94)		(0.94)
Autoencoder 1l (q = 05)	18.26	15.86	14.72	17.25
	(0.89*)	(0.89*)	(0.89*)	(0.89*)
Autoencoder 1l (q = 15)	24.56	27.39	22.94	26.17
	(0.86**)	(0.85**)	(0.86**)	(0.85**)
Autoencoder 1l (q = 30)	22.17	27.55	24.62	26.65
	(0.85**)	(0.84**)	(0.85**)	(0.83**)
Autoencoder 3l (q = 05)	19.24	22.33	21.18	19.49
	(0.87**)	(0.85**)	(0.87**)	(0.86**)
Autoencoder 3l (q = 15)	20.24	21.49	20.54	20.60
	(0.86**)	(0.86**)	(0.86**)	(0.85**)
Autoencoder 3l (q = 30)	23.97	20.29	21.60	20.45
	(0.86**)	(0.85**)	(0.86**)	(0.85**)
Autoencoder 5l (q = 05)	20.87	16.60	18.60	19.66
	(0.86**)	(0.86**)	(0.87**)	(0.86**)
Autoencoder 5l (q = 15)	15.15	14.34	15.71	14.00
	(0.86**)	(0.86**)	(0.87**)	(0.86**)
Autoencoder 5l (q = 30)	21.13	22.40	19.67	21.44
	(0.86**)	(0.85**)	(0.86**)	(0.85**)
Autoencoder 8l (q = 05)	20.82	18.27	16.17	17.18
	(0.88*)	(0.88*)	(0.89*)	(0.88*)
Autoencoder 8l (q = 15)	12.21	6.51	8.01	7.27
	(0.88*)	(0.92)	(0.90)	(0.91)
Autoencoder 8l (q = 30)	−3.75	−16.20	−11.91	−14.54
	(0.89)	(0.90)	(0.89)	(0.90)
Diffusion Maps (q = 05)	16.27	20.18	12.09	18.18
	(0.86**)	(0.86**)	(0.86**)	(0.86**)
Diffusion Maps (q = 15)	19.69	16.58	17.53	14.27
	(0.86**)	(0.86**)	(0.86**)	(0.86**)
Diffusion Maps (q = 30)	17.02	13.20	20.08	12.27
	(0.87**)	(0.89*)	(0.86**)	(0.88**)
ISOMAP (q = 05)	18.01	16.89	15.62	18.45
	(0.86**)	(0.86**)	(0.87**)	(0.86**)
ISOMAP (q = 15)	19.75	11.58	13.79	11.19
	(0.86**)	(0.86**)	(0.87**)	(0.86**)
ISOMAP (q = 30)	12.01	3.63	5.41	2.66
	(0.86**)	(0.85**)	(0.87**)	(0.86**)
LLE (q = 05)	14.10	5.25	9.71	5.27
	(0.87**)	(0.87**)	(0.87**)	(0.87**)
LLE (q = 15)	−11.38	−13.20	−18.14	−10.97
	(0.88*)	(0.88)	(0.89)	(0.88)
LLE (q = 30)	−1.39	−38.94	−11.37	−40.18
	(0.87**)	(0.87*)	(0.88*)	(0.87*)
PCA gauss. kernel (q = 05)	17.21	17.81	20.04	15.67
	(0.88*)	(0.88**)	(0.88*)	(0.88*)
PCA gauss. kernel (q = 15)	15.63	14.29	16.88	17.30
	(0.88*)	(0.88*)	(0.88*)	(0.88*)
PCA gauss. kernel (q = 30)	19.03	19.01	18.04	18.17
	(0.88**)	(0.88**)	(0.88**)	(0.88**)
PCA linear (q = 05)	17.05	15.66	15.05	14.73
	(0.90*)	(0.90*)	(0.90)	(0.90*)
PCA linear (q = 15)	18.79	19.39	16.35	14.59
	(0.88*)	(0.88*)	(0.89*)	(0.88*)
PCA linear (q = 30)	21.39	18.10	19.18	18.21
	(0.87**)	(0.87*)	(0.88*)	(0.87*)
PCA poly. kernel (q = 05)	18.98	16.48	15.09	14.71
	(0.89*)	(0.90*)	(0.89*)	(0.89*)
PCA poly. kernel (q = 15)	18.73	19.44	18.52	20.78
	(0.88*)	(0.88**)	(0.88*)	(0.88**)
PCA poly. kernel (q = 30)	22.11	20.63	19.63	21.22
	(0.87**)	(0.87**)	(0.88**)	(0.87**)

(*continued on next page*)

Table 3 (*continued*).

PCA quadratic (q = 05)	40.33***	**46.67****	41.31***	**48.50*****
	(0.84**)	(0.79***)	(0.84**)	(0.79***)
PCA quadratic (q = 15)	20.37	33.39	17.63	34.40
	(0.91)	(0.86*)	(0.91)	(0.86*)
PCA quadratic (q = 30)	24.90	26.61	28.50*	26.91
	(0.87**)	(0.89**)	(0.87**)	(0.90**)
PCA squared (q = 05)	**46.84*****	46.30**	**45.90*****	47.62**
	(**0.80****)	(**0.76***)	(**0.81****)	(**0.76***)
PCA squared (q = 15)	24.28	33.35*	23.02	32.54*
	(0.87*)	(0.87*)	(0.87*)	(0.88*)
PCA squared (q = 30)	28.25*	27.59	26.46*	27.35
	(0.86**)	(0.88**)	(0.86**)	(0.88**)

Note: The table shows log predictive likelihoods (LPLs) with root mean squared errors (RMSEs) in parentheses below. The first (red shaded) entry gives the actual LPL and RMSE scores of our benchmark (an autoregressive (AR) model with constant parameters and a Minnesota prior). Asterisks indicate statistical significance for each model relative to the benchmark at the 1% (***), 5% (**) and 10% (*) significance levels. Since the large ARX model with time-varying parameters would feature 273 period-specific coefficients and is computationally intractable, we assume that the TVPs feature a factor structure (with three factors) to reduce the dimension of the state space (see Section B of the Online Appendix and Chan et al., 2020).

Table 4
Summary of best performing models and comparision to small-scale unsupervised ARX models.

Specification	const. (MIN)	const. (HS)	TVP (MIN)	TVP (HS)
One-month-ahead				
AR	−329.28			
	(1.24)			
Autoencoder 1l (q = 30)	38.19***	29.92***	35.48***	31.39***
	(0.81***)	(0.84***)	(0.80***)	(0.83***)
Small ARX based on	1.39	−2.02	1.97	−2.27
Autoencoder 1l (q = 30)	(0.98)	(0.99)	(0.98)	(0.99)
One-quarter-ahead				
AR	−322.66			
	(1.27)			
PCA quadratic (q = 05)	40.33***	46.67**	41.31***	48.50***
	(0.84**)	(0.79***)	(0.84**)	(0.79***)
PCA squared (q = 05)	46.84***	46.30**	45.90***	47.62**
	(0.80**)	(0.76***)	(0.81**)	(0.76***)
Small ARX based on	38.20**	43.73**	39.41**	43.28**
PCA quadratic (q = 5)	(1.05)	(0.93)	(1.04)	(0.93)
Small ARX based on	36.25**	42.90**	38.87**	44.25**
PCA squared (q = 5)	(1.09)	(0.95)	(1.09)	(0.95)

Note: The table shows log predictive likelihoods (LPLs) with root mean squared errors (RMSEs) in parentheses below. While the first entry denotes the benchmark (shaded in red), other entries contrast the best performing model(s) as presented in the Tables 2 and 3 with ARX models. For these ARX specifications the best performing model solely serves as an unsupervised variable selection device. Conditional on the latent factors of this best performing dimension reduction technique, we always include the top-five correlated variables as covariates alongside the lags of inflation (see Figure C.1 of the Online Appendix). Asterisks indicate statistical significance for each model relative to the benchmark at the 1% (***), 5% (**) and 10% (*) significance levels.

The asterisks indicate whether the variances are significantly different from one. For a few models, this is the case (especially if we assume constancy of the parameters), but if we allow for TVPs, there are only a handful of cases left. This, however, strongly depends on the shrinkage prior adopted. Turning to the autocorrelation of the normalized shocks reveals that these are mostly close to zero and never statistically significantly different from zero.

Comparing sophisticated to simple dimension reduction methods suggests no discernible differences in model calibration. In principle, approaches based on linear PCs yield normalized forecast errors with similar statistical properties to those obtained using more sophisticated dimension reduction techniques.

The discussion above might mask important differences in the calibration of different parts of the predictive distribution. We now turn to a deeper analysis of the one-month-ahead predictive distribution of the two best performing models vis-á-vis the benchmark: the Autoencoder 1l ($q = 30$) and PCA squared ($q = 5$). This analysis is based on visual inspection of the normalized forecast errors (left panel of Fig. 3), a histogram of the PITs (middle panel of Fig. 3) and the visual diagnostic of the empirical cumulative density function proposed in Rossi and Sekhposyan (2019) (right panel of Fig. 3).

Table 5
Test statistics of one-month-ahead probability integral transformations.

Specification	const. (MIN)			const. (HS)			TVP (MIN)			TVP (HS)		
	Mean	Variance	AR(1) coef.	Mean	Variance	AR(1) coef.	Mean	Variance	AR(1) coef.	Mean	Variance	AR(1) coef.
AR	0.018	1.171	0.063	0.035	1.182	0.067	0.024	1.165	0.032	0.038	1.189	0.063
Large ARX	0.036	1.166	0.086	0.029	1.198	0.061				0.026	1.195	0.069
Autoencoder 1l (q = 05)	0.028	1.182	0.082	0.031	1.194	0.068	0.024	1.188	0.081	0.034	1.191	0.063
Autoencoder 1l (q = 15)	0.020	1.176	0.067	0.029	1.203	0.054	0.007	1.155	0.066	0.030	1.199	0.054
Autoencoder 1l (q = 30)	0.010	1.293**	0.010	0.028	1.336**	0.002	0.005	1.307**	−0.013	0.023	1.317**	−0.004
Autoencoder 3l (q = 05)	0.030	1.175	0.083	0.038	1.191	0.063	0.030	1.183	0.067	0.035	1.196	0.067
Autoencoder 3l (q = 15)	0.045	1.187	0.084	0.044	1.200	0.059	0.029	1.193	0.062	0.045	1.203	0.057
Autoencoder 3l (q = 30)	−0.005	1.277*	0.050	0.009	1.315**	0.018	−0.041	1.343*	0.061	0.005	1.357**	0.027
Autoencoder 5l (q = 05)	0.034	1.181	0.084	0.036	1.189	0.065	0.031	1.183	0.077	0.039	1.192	0.062
Autoencoder 5l (q = 15)	0.040	1.191	0.068	0.057	1.220	0.041	0.032	1.196	0.056	0.058	1.218	0.035
Autoencoder 5l (q = 30)	0.062	1.225	0.074	0.067	1.230	0.058	0.039	1.244	0.059	0.066	1.233	0.046
Autoencoder 8l (q = 05)	0.034	1.190	0.088	0.039	1.193	0.068	0.034	1.167	0.081	0.036	1.201	0.064
Autoencoder 8l (q = 15)	0.038	1.174	0.079	0.033	1.201	0.075	0.030	1.193	0.090	0.039	1.189	0.071
Autoencoder 8l (q = 30)	0.032	1.164	0.070	0.027	1.187	0.048	0.025	1.152	0.072	0.026	1.180	0.045
Diffusion Maps (q = 05)	0.025	1.193	0.079	0.023	1.212	0.070	0.012	1.166	0.070	0.026	1.198	0.067
Diffusion Maps (q = 15)	0.037	1.192	0.079	0.031	1.211	0.068	0.027	1.188	0.075	0.031	1.210	0.064
Diffusion Maps (q = 30)	0.032	1.194	0.083	0.034	1.212	0.059	0.029	1.187	0.070	0.033	1.190	0.054
ISOMAP (q = 05)	0.024	1.170	0.085	0.026	1.197	0.057	0.020	1.175	0.082	0.031	1.178	0.060
ISOMAP (q = 15)	0.027	1.180	0.086	0.027	1.189	0.058	0.024	1.181	0.080	0.021	1.194	0.059
ISOMAP (q = 30)	0.031	1.195	0.079	0.026	1.194	0.068	0.026	1.193	0.085	0.031	1.188	0.065
LLE (q = 05)	0.029	1.196	0.083	0.025	1.191	0.071	0.019	1.177	0.085	0.026	1.181	0.071
LLE (q = 15)	0.027	1.189	0.083	0.025	1.206	0.067	0.020	1.185	0.084	0.022	1.196	0.066
LLE (q = 30)	0.031	1.194	0.080	0.034	1.261*	0.045	0.013	1.194	0.086	0.033	1.247*	0.040
PCA gauss. kernel (q = 05)	0.027	1.174	0.081	0.035	1.190	0.066	0.025	1.194	0.084	0.035	1.175	0.072
PCA gauss. kernel (q = 15)	0.032	1.178	0.087	0.041	1.189	0.067	0.030	1.192	0.080	0.036	1.205	0.066
PCA gauss. kernel (q = 30)	0.032	1.178	0.078	0.034	1.203	0.069	0.023	1.170	0.083	0.038	1.182	0.065
PCA linear (q = 05)	0.033	1.190	0.083	0.042	1.208	0.075	0.023	1.192	0.091	0.041	1.183	0.063
PCA linear (q = 15)	0.034	1.177	0.083	0.037	1.192	0.070	0.026	1.172	0.083	0.038	1.187	0.064
PCA linear (q = 30)	0.031	1.185	0.081	0.035	1.210	0.067	0.025	1.176	0.087	0.030	1.187	0.063
PCA poly. kernel (q = 05)	0.029	1.170	0.081	0.036	1.193	0.067	0.029	1.178	0.081	0.037	1.189	0.061
PCA poly. kernel (q = 15)	0.028	1.201	0.078	0.034	1.193	0.061	0.034	1.187	0.077	0.031	1.192	0.068
PCA poly. kernel (q = 30)	0.031	1.166	0.090	0.043	1.185	0.066	0.029	1.182	0.080	0.042	1.199	0.059
PCA quadratic (q = 05)	0.029	1.071	0.044	0.031	1.070	0.023	0.034	1.055	0.026	0.032	1.052	0.016
PCA quadratic (q = 15)	0.032	1.178	0.069	0.008	1.190	0.063	0.021	1.158	0.056	0.003	1.181	0.052
PCA quadratic (q = 30)	0.028	1.177	0.086	−0.003	1.251*	0.085	0.032	1.163	0.061	−0.004	1.232	0.090
PCA squared (q = 05)	0.042	1.071	0.035	0.035	1.056	0.021	0.040	1.030	0.027	0.031	1.061	0.013
PCA squared (q = 15)	0.031	1.174	0.070	0.006	1.173	0.049	0.033	1.145	0.041	0.007	1.157	0.043
PCA squared (q = 30)	0.037	1.172	0.079	−0.002	1.203	0.069	0.043	1.148	0.062	−0.008	1.224	0.080

Note: This table summarizes the normalized forecast errors, which are obtained with probability integral transforms (PITs). Similar to Clark (2011) we show the mean, the variance and the AR(1) coefficient of the normalized forecast errors. Given a well-calibrated model (i.e. the null-hypothesis), normalized forecast errors should have zero mean, a variance of one and experience no autocorrelation. These conditions are tested separately: 1) To test for a zero mean we compute the p-values with a Newey-West variance (with five lags). 2) To test for a unit variance we regress the squared normalized forecast errors on an intercept and allow for a Newey-West variance (with three lags). 3) To test for no autocorrelation we obtain the p-values with an AR(1) model that features an unconditional mean and heteroskedasticity-robust standard errors. Asterisks indicate statistical significance for each model at the 1% (***), 5% (**) and 10% (*) significance levels.

Recall that, under correct specification, the PITs should be iid uniformly, and the normalized forecast errors should be iid standard normally distributed, respectively.

The left panel of the figure indicates that for both models under consideration, normalized forecast errors are centered on zero, display little serial correlation, and a variance close to one (with the Autoencoder generating slightly more spread-out normalized forecast errors). In some periods, normalized forecast errors depart significantly from the standard normal distribution (i.e., the corresponding observations lie outside the 95% confidence intervals). But in general, and for both models (and the benchmark), model calibration seems adequate. Next, we focus on the histogram in the middle panel of Fig. 3 (which includes 95% confidence intervals). From this figure, we learn that both models are well calibrated with some tendency to overestimate the upper tail risk. Finally, considering the right panel shows that all models appear to be well-calibrated, with most observations clustered around the 45-degree lines and not a single observation being outside the 95% confidence intervals.

4.5. A note on the pandemic

To the detriment of linear modeling techniques, the COVID-19 pandemic caused severe outliers for several of the time series we include in our dataset. Following the recent literature (e.g., Clark et al., 2021; Coulombe et al., 2021; Huber et al., 2020) that advocates using non-linear and non-parametric modeling techniques in turbulent times, we briefly investigate whether the non-linear dimension reduction techniques proposed in this paper yield more precise inflation forecasts during the pandemic.

Fig. 4 depicts the differences in LPLs for the period 2020:01 to 2020:08. For illustrative purposes, we only consider the models with 30 factors.[9]

The figure provides a few interesting insights. First, we observe that in March 2020, models based on the Autoencoder improve upon the benchmark, irrespective of the prior and regression specification adopted. This finding is less pronounced for the other techniques in the constant parameter case. Comparing the performance of the constant parameter and the TVP regression models reveals that, irrespective of the prior, allowing for time variation in the parameters improves density forecasts during the pandemic. This finding is consistent with findings in, e.g., Huber et al. (2020), which show that flexible models improve upon linear models during the pandemic due to increases in the predictive variance.

[9] The findings for the other factors are very similar and available from the corresponding author upon request.

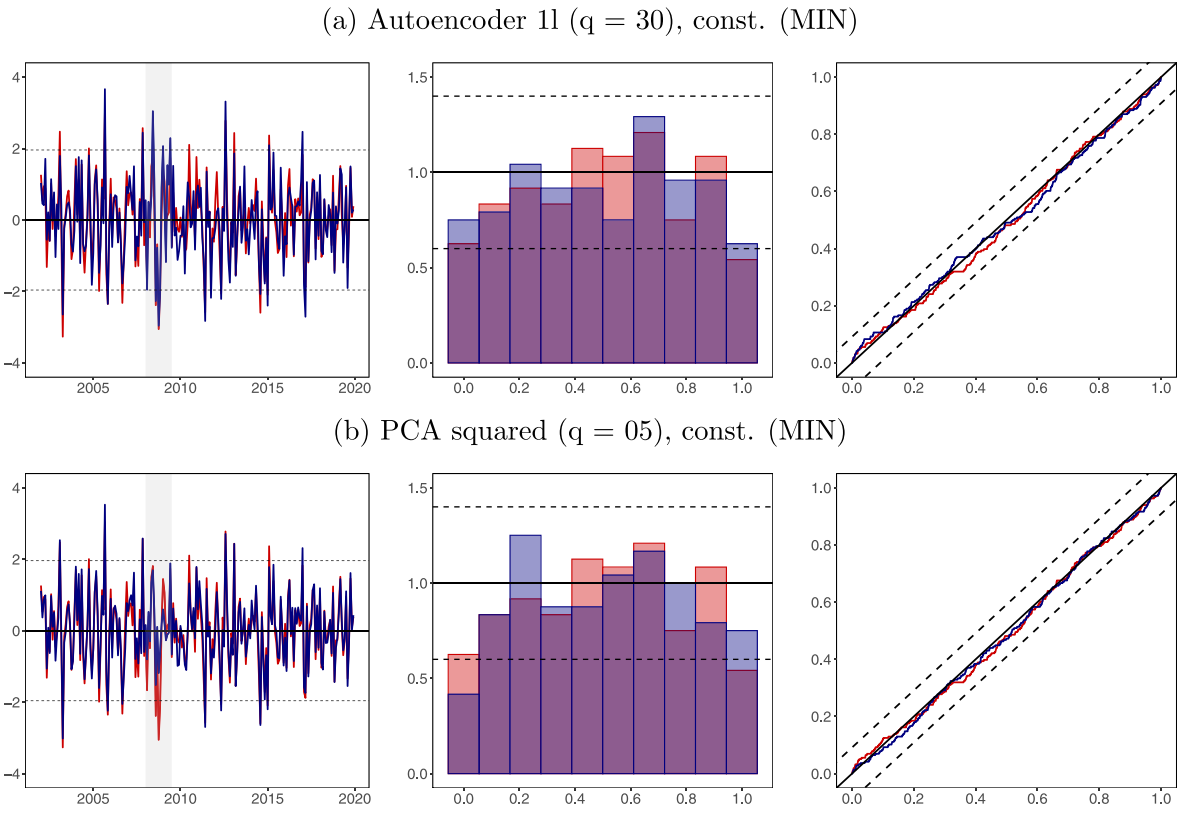

Fig. 3. Detailed analysis of one-month-ahead predictive distributions for selected models.
Note: This figure shows the evolution of normalized forecast errors for the one-month-ahead horizon in the left panel, the histogram of the probability integral transforms (PITs) in the middle panel and the empirical cumulative density function of the PITs in the right panel. If a model is correctly specified the PITs are standard uniformly distributed and the normalized forecast errors standard normally distributed. This theoretically correct specification is indicated by the black lines, with the dashed lines referring to the respective 95% confidence interval. In red we present the results of the benchmark, whereas in blue we indicate the respective model. In the left panels the light gray shaded areas refer to the global financial crisis.

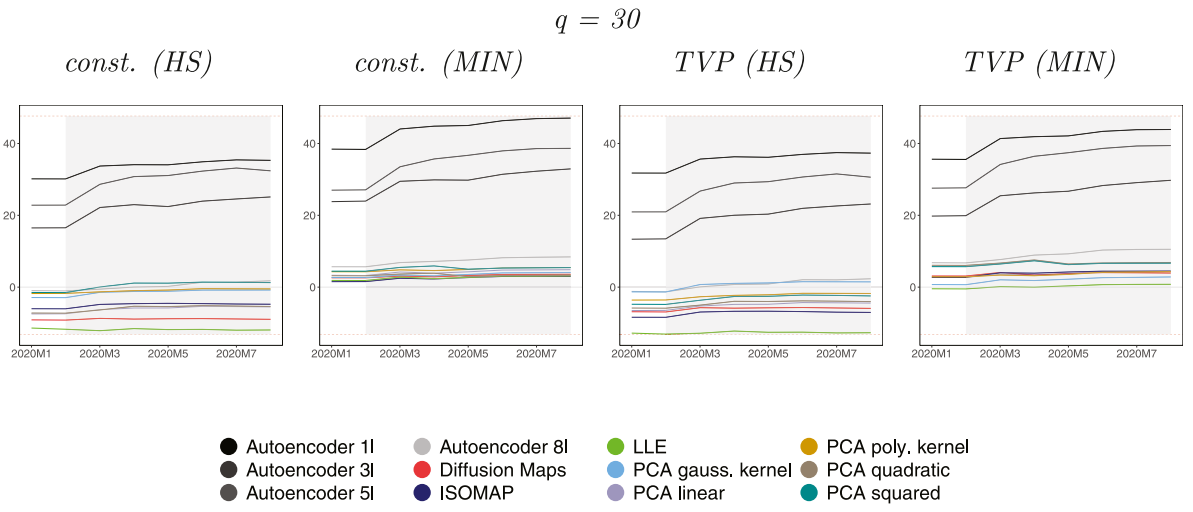

Fig. 4. Evolution of one-month-ahead cumulative log predictive likelihoods (LPLs) against the benchmark for the COVID-19 pandemic.
Note: The initial value in 2020:01 also takes into account the density forecast performance in the previous hold-out periods (ranging from 2002:01 to 2019:12). The red dashed lines refer to the maximum/minimum Bayes factor over the full hold-out sample. The light gray shaded areas indicate the periods of the COVID-19 pandemic.

4.6. Dynamic model learning based on density forecast performance

In the previous subsection and Section A of the Online Appendix, we provide some evidence that model performance varies considerably over time (see Figure A.3). The key implication is that non-linear compression techniques (and time-varying parameters) might be useful during turbulent times, whereas forecast evidence is less pronounced in normal times. This subsection asks whether dynamically combining models further improves predictive accuracy.

After having obtained the predictive densities of y_{t+h} for the different dimension reduction techniques and model specifications, the goal is to exploit the advantages of both linear and non-linear approaches. This is achieved by combining models in a model pool such that better-performing models over certain periods receive larger weights while inferior models are subsequently down-weighted. The literature on forecast combinations suggests several different weighting schemes, ranging from simply averaging over all models (see, e.g., Berg & Henzel, 2015; Clark & McCracken, 2010; Hall & Mitchell, 2007; Hendry & Clements, 2004) to estimating weights based on the models' performances according to the minimization of an objective or loss function (see, e.g., Conflitti et al., 2015; Geweke & Amisano, 2011; Hall & Mitchell, 2007; Pettenuzzo & Ravazzolo, 2016; Timmermann, 2006) or according to the posterior probabilities of the predictive densities (see, e.g., Beckmann et al., 2020; Koop & Korobilis, 2012; Raftery et al., 2010). More recent approaches set up separate state space models that assume sophisticated law of motions for the weights associated with each predictive distribution (Billio et al., 2013; McAlinn & West, 2019; Pettenuzzo & Ravazzolo, 2016). While being elegant and having the advantage of incorporating all available sources of uncertainty (i.e., control for estimation uncertainty in the weights), these approaches are computationally cumbersome if the number of models to be combined is large.

Since our model space is huge, we use computationally efficient approximations to dynamically combine models. Our approach builds on combining predictive densities according to their posterior probabilities. This is referred to as Bayesian model averaging (BMA). The resulting weights are able to reflect the predictive power of each model for the respective periods. Dynamic model averaging (DMA), as specified by Raftery et al. (2010), extends the approach by adding a discount (or *forgetting*) factor to control for a model's forecasting performance in the recent past. The 'recent past' is determined by the discount factor, with higher values attaching greater importance to past forecasting performances of the model and lower values gradually ignoring results of past predictive densities. Similar to Raftery et al. (2010), Koop and Korobilis (2012) and Beckmann et al. (2020), we apply DMA to combine the predictive densities of our various models. These methods do not require computationally intensive MCMC or sequential Monte Carlo techniques and are thus fast and easy to implement.

DMA works as follows. Let $\boldsymbol{\varrho}_{t+h|t} = (\varrho_{t+h|t,1}, \ldots, \varrho_{t+h|t,J})'$ denote a set of weights for J competing models at time $t+h$ given all available information up to time t. These (horizon-specific) weights vary over time and depend on the recent predictive performance of the model according to:

$$\varrho_{t+h|t,j} = \frac{\varrho_{t|t,j}^{\delta}}{\sum_{l=1}^{J} \varrho_{t|t,l}^{\delta}},$$

$$\varrho_{t+h|t+h,j} = \frac{\varrho_{t+h|t,j}\; p_j(y_{t+h}|y_{1:t})}{\sum_{l=1}^{J} \varrho_{t+h|t,l}\; p_l(y_{t+h}|y_{1:t})}$$

where $p_j(y_{t+h}|y_{1:t})$ denotes the h-month-ahead predictive distribution of model j evaluated at y_{t+h} and $\delta \in (0, 1]$ denotes a forgetting factor close to one. Intuitively speaking, the first equation is a prediction of the weights based on all available information up to time t while the second equation shows how the weights get updated if new data flows in.

In our empirical work we set $\delta = 0.97$.[10] Notice that if $\delta = 1$, we obtain standard BMA weights while $\delta = 0$ would imply that the weights depend exclusively on the forecasting performance in the last period.

4.7. Forecasting performance of predictive combinations from dynamic model learning

Weights obtained by combining models according to their predictive power convey useful information about the adequacy of each model over time. To get a comprehensive picture of the effects of different model modifications, we combine our models and model specifications in various ways.

Table 6 presents the forecasting results when we use DMA to combine models. Again, all models are benchmarked to the AR model with constant parameters and the Minnesota prior. The first row depicts the relative performance of the single best-performing model for the chosen time horizon.

The table can be understood as follows. Each entry includes *all* dimension reduction techniques. The rows define whether the model space includes all factors $q \in \{5, 15, 30\}$ or whether we combine models with a fixed number of factors exclusively. The columns refer to model spaces that include only the constant parameter, time-varying parameter, or both specifications in the respective model pool. Since we also discriminate between two competing priors, we consider model weights conditioning on either the horseshoe or the Minnesota prior or average across both prior specifications (the first upper part of the table with {HS, MIN}).

Across the two forecast horizons considered, we find pronounced accuracy improvements for point and density forecasts relative to the AR model. We find no accuracy gains for both horizons when we benchmark the different combination strategies to the single best-performing model. Differences in terms of LPLs are, however, relatively small. This suggests that while the best performing model (i.e., a constant parameter regression with factors

[10] Koop and Korobilis (2013) find robust results over the interval [0.95,1]. After optimizing over this set of parameter values we choose $\delta = 0.97$.

Table 6
Forecast performance of predictive combinations.

Specification		One-month-ahead			One-quarter-ahead		
Prior	Combination	const.	TVP	{const., TVP}	const.	TVP	{const., TVP}
Single best performing model		38.19 (0.81)				48.50 (0.79)	
{HS, MIN}	q = {05, 15, 30}	33.40 (0.83)	31.53 (0.82)	32.73 (0.82)	45.88 (0.79)	46.65 (0.79)	46.32 (0.79)
	q = 05	7.97 (0.93)	8.68 (0.92)	8.40 (0.92)	46.49 (0.78)	47.51 (0.78)	47.09 (0.78)
	q = 15	9.84 (0.94)	14.81 (0.90)	13.28 (0.91)	40.45 (0.84)	40.96 (0.83)	40.83 (0.83)
	q = 30	34.80 (0.83)	33.45 (0.82)	34.37 (0.82)	33.25 (0.86)	32.82 (0.87)	33.05 (0.86)
HS	q = {05, 15, 30}	26.26 (0.85)	29.17 (0.84)	28.07 (0.84)	47.29 (0.77)	48.02 (0.77)	47.75 (0.77)
	q = 05	6.68 (0.94)	6.74 (0.94)	6.74 (0.94)	**47.59** (**0.76**)	**48.68** (**0.76**)	**48.30** (**0.76**)
	q = 15	8.03 (0.94)	9.02 (0.94)	8.68 (0.94)	42.57 (0.83)	42.65 (0.82)	42.73 (0.82)
	q = 30	28.05 (0.85)	31.05 (0.84)	29.89 (0.84)	34.82 (0.85)	33.95 (0.86)	34.40 (0.85)
MIN	q = {05, 15, 30}	35.96 (0.82)	32.23 (0.81)	34.65 (0.81)	43.37 (0.82)	43.57 (0.82)	43.50 (0.82)
	q = 05	8.62 (0.92)	10.10 (0.91)	9.57 (0.91)	44.80 (0.82)	44.64 (0.82)	44.74 (0.82)
	q = 15	10.98 (0.94)	17.02 (0.89)	15.40 (0.90)	26.16 (0.86)	25.92 (0.86)	26.01 (0.86)
	q = 30	**37.27** (**0.82**)	**34.60** (**0.80**)	**36.46** (**0.81**)	23.63 (0.86)	26.12 (0.87)	25.27 (0.87)

Note: The first (grey shaded) row states the results of the single best performing model as presented in the previous chapter for each forecast horizon benchmarked to the AR model with constant parameters and the Minnesota (MIN) prior. All other rows show the relative results for the combinations of the different dimension reduction techniques according to the specification stated in the rows and columns headers. For example, the entry in row {HS, MIN}, $q = \{5, 15, 30\}$ and column const. combines all models estimated with constant parameters, the HS prior, the MIN prior, 5, 15 and 30 factors. Entries denote the differences in log predictive likelihoods (LPLs) with relative root mean squared errors (RMSEs) in parantheses benchmarked against the AR model with constant parameters and the MIN prior.

obtained through the Autoencoder) is hard to beat, one can effectively reduce model and specification uncertainty and thus obtain competitive forecasts without relying on a single model.

Comparing whether restricting the model a priori improves predictions yields mixed insights. For the one-month-ahead horizon, we observe that pooling over models that use our variant of the Minnesota prior yields more favorable forecasts than a pooling strategy that uses both priors or the horseshoe only. When we pool over constant and TVP regressions, we find small decreases in predictive accuracy relative to a model pool that only includes constant parameter regressions.

Turning to one-quarter-ahead forecasts yields a similar picture. Using a large pool of models generally leads to slightly less precise forecasts. For higher-order forecasts, our results suggest that pooling models that use the horseshoe yield higher LPLs. When we compare the different regression specifications, we find that integrating out the uncertainty with respect to whether parameters should be time-varying yields forecasts that are very similar to the strategy that only pools over constant parameter models.

In general, the differences in predictive performance across the DMA-based averaging schemes are small. Hence, as a general suggestion, we recommend applying DMA and using the most exhaustive model space available (i.e., including both priors, the different number of factors, and TVP and constant parameter regressions).

To investigate which model receives substantial posterior weight over time, Fig. 5 depicts the weights associated with the one-month-ahead LPLs over the hold-out period. Panel (a) displays the weight placed on models that allow for TVP, panel (b) shows the weight attached to the different number of factors, and panel (c) shows the weight attached to each model. These weights are obtained using the full model space (i.e., priors, TVP, constant parameter regressions, and any number of factors). The weight placed on TVP specifications, for instance, is then simply obtained by summing up the weights associated with the different models that feature TVPs.

Starting with the top panel of the figure, we observe that appreciable model weight is placed on constant parameter models during the beginning of the sample. In the middle of 2006, this changes, and DMA places increasing posterior mass on models that allow for time-variation

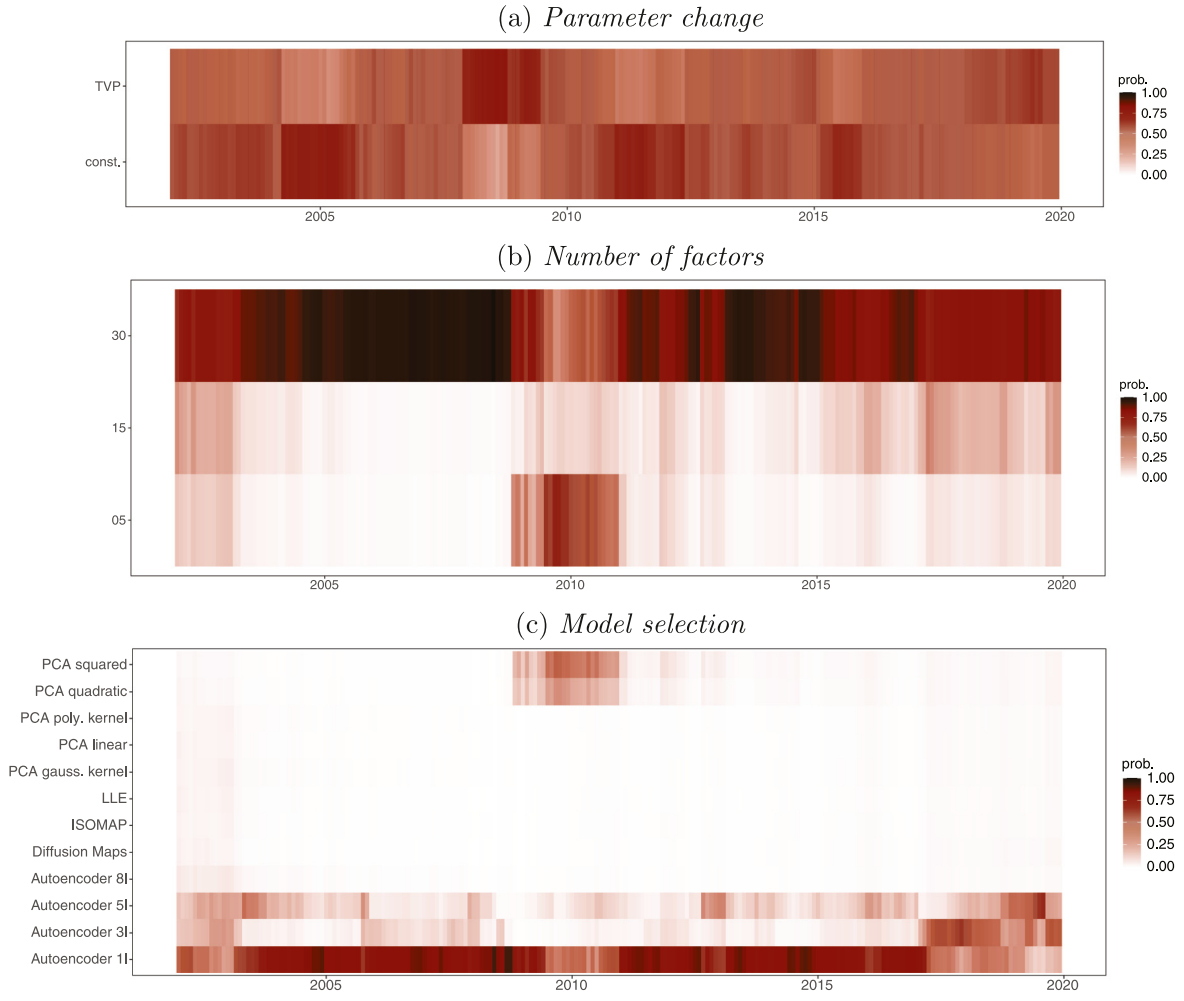

Fig. 5. Evolution of the weights determined by DMA for one-month-ahead cumulative log predictive likelihoods (LPLs).

in the parameters. From the beginning of 2007 to the onset of the financial crisis, we see that the weight on TVP models somewhat decreases. We again experienced a pronounced increase in posterior weight towards TVP regression during the financial crisis. In that period, constant parameter models only played a limited role in forming inflation forecasts. With few exceptions, the remainder of the hold-out period is characterized by evenly distributed posterior mass across constant and TVP regressions.

The middle panel of Fig. 5 shows that DMA places increasing posterior mass on models with a large number of factors during the period before the global financial crisis. We observe that models with a smaller number of factors obtain substantial model weight during the recession and the immediate period afterward. This suggests that if the number of factors is small, our dimension reduction techniques soak up information that might be useful during a recession (such as sharp changes in $\boldsymbol{X}$). If the number of factors becomes large, this information is extracted as well but (potentially) reflected by more factors that display less pronounced changes. In the period after the global financial crisis, we again find a large number of factors retrieving substantial posterior weight.

The bottom panel (panel (c)) of Fig. 5 provides information on how much weight is allocated to models that exploit non-linear dimension reduction techniques. This figure corroborates our full sample findings: the Autoencoder performs extremely well and dominates our pool of models. However, this statement is not valid during the global financial crisis. We observe that models based on PCA squared and PCA quadratic feature large weights during that period. We also find that linear techniques (PCA linear) and other non-linear techniques (PCA with a Gaussian kernel, LLE, ISOMAP, diffusion maps) retrieve almost no posterior weight over time.

Summing up this discussion, the single best performing model (the Autoencoder) is hard to beat when dynamically combining models. However, this comparison is unfair since the researcher does not have this information at her disposal. Hence, combining models helps to integrate this uncertainty by producing forecasts that are close to the single best performing model but, at the cost of higher computational costs, without the necessity of knowing the strongest single model specification.

5. Closing remarks

In macroeconomics, most researchers compress information using linear methods such as principal components to summarize information embodied in huge datasets in forecasting applications efficiently. Machine learning techniques describing large datasets with relatively few latent factors have gained relevance in the last years in various areas. This paper shows that using such approaches potentially improves real-time inflation forecasts for a wide range of competing model specifications. Our findings indicate that point forecasts of simpler models are hard to beat (especially at the one-month-ahead horizon). However, we find that more sophisticated modeling techniques that rely on non-linear dimension reduction do particularly well for density forecasts. Among all the techniques considered, our results suggest that the Autoencoder, a particular form of a deep neural network, produces the most precise inflation forecasts (both in terms of point and density predictions). The large battery of competing models gives rise to substantial model uncertainty. We address this issue by using dynamic model averaging to dynamically weight different models, dimension reduction methods, and priors; doing so yields almost as accurate forecasts as those obtained from the single best performing models.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Supplementary data

The online appendix provides additional empirical results such as the factors obtained from the different nonlinear dimension reduction techniques, forecasting results for subgroups of the data and predictive accuracy over time. Moreover, it provides details on the posterior simulation algorithms and the dataset.

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.ijforecast.2022.03.002.

References

Allaire, J., & Chollet, F. (2019). Keras: R interface to 'Keras'. R package version 2.2.5.0.

Andreini, P., Izzo, C., & Ricco, G. (2020). Deep dynamic factor models. arXiv preprint arXiv:2007.11887.

Angerer, P., Haghverdi, L., Büttner, M., Theis, F. J., Marr, C., & Buettner, F. (2016). Destiny: diffusion maps for large-scale single-cell data in R. *Bioinformatics*, *32*(8), 1241–1243.

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

Bai, J., & Ng, S. (2008). Forecasting economic time series using targeted predictors. *Journal of Econometrics*, *146*(2), 304–317.

Beckmann, J., Koop, G., Korobilis, D., & Schüssler, R. A. (2020). Exchange rate predictability and dynamic Bayesian learning. *Journal of Applied Econometrics*, *35*(4), 410–421.

Belmonte, M., Koop, G., & Korobilis, D. (2014a). Hierarchical shrinkage in time-varying coefficient models. *Journal of Forecasting*, *33*(1), 80–94.

Belmonte, M. A., Koop, G., & Korobilis, D. (2014b). Hierarchical shrinkage in time-varying parameter models. *Journal of Forecasting*, *33*(1), 80–94.

Berg, T. O., & Henzel, S. R. (2015). Point and density forecasts for the euro area using Bayesian VARs. *International Journal of Forecasting*, *31*(4), 1067–1095.

Bernanke, B. S., Boivin, J., & Eliasz, P. (2005). Measuring the effects of monetary policy: A factor-augmented vector autoregressive (FAVAR) approach. *Quarterly Journal of Economics*, *120*(1), 387–422.

Billio, M., Casarin, R., Ravazzolo, F., & Van Dijk, H. K. (2013). Time-varying combinations of predictive densities using nonlinear filtering. *Journal of Econometrics*, *177*(2), 213–232.

Carriero, A., Clark, T. E., & Marcellino, M. (2015). Bayesian VARs: specification choices and forecast accuracy. *Journal of Applied Econometrics*, *30*(1), 46–73.

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

Chakraborty, C., & Joseph, A. (2017). *Machine learning at central banks*: *Working papers 674*, Bank of England.

Chan, J. C. (2017). The stochastic volatility in mean model with time-varying parameters: An application to inflation modeling. *Journal of Business & Economic Statistics*, *35*(1), 17–28.

Chan, J. C. (2021). Minnesota-type adaptive hierarchical priors for large Bayesian VARs. *International Journal of Forecasting*, *37*(3), 1212–1226.

Chan, J. C., Clark, T. E., & Koop, G. (2018). A new model of inflation, trend inflation, and long-run inflation expectations. *Journal of Money, Credit and Banking*, *50*(1), 5–53.

Chan, J. C., Eisenstat, E., & Strachan, R. W. (2020). Reducing the state space dimension in a large TVP-var. *Journal of Econometrics*, *218*(1), 105–118.

Clark, T. (2011). Real-time density forecasts from BVARs with stochastic volatility. *Journal of Business & Economic Statistics*, *29*, 327–341.

Clark, T. E., Huber, F., Koop, G., Marcellino, M., & Pfarrhofer, M. (2021). *Tail forecasting with multivariate bayesian additive regression trees*: *Federal Reserve Bank of Cleveland Working Paper, No. 21-08*.

Clark, T. E., & McCracken, M. W. (2010). Averaging forecasts from VARs with uncertain instabilities. *Journal of Applied Econometrics*, *25*(1), 5–29.

Clark, T. E., & Ravazzolo, F. (2015). Macroeconomic forecasting performance under alternative specifications of time-varying volatility. *Journal of Applied Econometrics*, *30*(4), 551–575.

Coifman, R. R., & Lafon, S. (2006). Diffusion maps. *Applied and Computational Harmonic Analysis*, *21*(1), 5–30.

Coifman, R. R., Lafon, S., Lee, A. B., Maggioni, M., Nadler, B., Warner, F., & Zucker, S. W. (2005). Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. *Proceedings of the National Academy of Sciences*, *102*(21), 7426–7431.

Conflitti, C., De Mol, C., & Giannone, D. (2015). Optimal combination of survey forecasts. *International Journal of Forecasting*, *31*(4), 1096–1103.

Coulombe, P. G., Leroux, M., Stevanovic, D., & Surprenant, S. (2019). *How is machine learning useful for macroeconomic forecasting?*: *CIRANO working papers 2019s-22*, CIRANO.

Coulombe, P. G., Marcellino, M., & Stevanović, D. (2021). Can machine learning catch the Covid-19 recession? *National Institute Economic Review*, *256*, 71–109.

Crawford, L., Flaxman, S. R., Runcie, D. E., & West, M. (2019). Variable prioritization in nonlinear black box methods: A genetic association case study. *The Annals of Applied Statistics*, *13*(2), 958.

Crawford, L., Wood, K. C., Zhou, X., & Mukherjee, S. (2018). Bayesian approximate kernel regression with variable selection. *Journal of the American Statistical Association*, *113*(524), 1710–1721.

Croushore, D. (2011). Frontiers of real-time data analysis. *Journal of Economic Literature*, *49*(1), 72–100.

D'Agostino, A., Gambetti, L., & Giannone, D. (2013). Macroeconomic forecasting and structural change. *Journal of Applied Econometrics*, *28*(1), 82–101.

De Mol, C., Giannone, D., & Reichlin, L. (2008). Forecasting using a large number of predictors: Is Bayesian shrinkage a valid alternative to principal components? *Journal of Econometrics*, *146*(2), 318–328.

Diebold, F. X., & Mariano, R. S. (1995). Comparing predictive accuracy. *Journal of Business & Economic Statistics*, *13*(3), 253–263.

Diedrich, H., & Abel, D. M. (2012). Lle: Locally linear embedding. R package version 1.1.

Exterkate, P., Groenen, P. J., Heij, C., & van Dijk, D. (2016). Nonlinear forecasting with many predictors using kernel ridge regression. *International Journal of Forecasting*, *32*(3), 736–753.
Feng, G., He, J., & Polson, N. G. (2018). Deep learning for predicting asset returns. arXiv preprint arXiv:1804.09314.
Gallant, A. R., & White, H. (1992). On learning the derivatives of an unknown mapping with multilayer feedforward networks. *Neural Networks*, *5*(1), 129–138.
Geweke, J., & Amisano, G. (2011). Optimal prediction pools. *Journal of Econometrics*, *164*(1), 130–141.
Giannone, D., Lenza, M., & Primiceri, G. E. (2015). Prior selection for vector autoregressions. *The Review of Economics and Statistics*, *97*(2), 436–451.
Giovannelli, A. (2012). *Nonlinear forecasting using large datasets: Evidences on US and Euro area economies*: *CEIS research paper 255*, Tor Vergata University, CEIS.
Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.
Hall, S. G., & Mitchell, J. (2007). Combining density forecasts. *International Journal of Forecasting*, *23*(1), 1–13.
Hauzenberger, N., Huber, F., Koop, G., & Onorante, L. (2019). Fast and flexible Bayesian inference in time-varying parameter regression models. arXiv preprint arXiv:1910.10779.
Heaton, J. (2008). *Introduction to neural networks with java*. Heaton Research, Inc.
Heaton, J. B., Polson, N. G., & Witte, J. H. (2017). Deep learning for finance: deep portfolios. *Applied Stochastic Models in Business and Industry*, *33*(1), 3–12.
Hendry, D. F., & Clements, M. P. (2004). Pooling of forecasts. *The Econometrics Journal*, *7*(1), 1–31.
Huang, G.-B. (2003). Learning capability and storage capacity of two-hidden-layer feedforward networks. *IEEE Transactions on Neural Networks*, *14*(2), 274–281.
Huber, F., Koop, G., & Onorante, L. (2021). Inducing sparsity and shrinkage in time-varying parameter models. *Journal of Business & Economic Statistics*, *39*(3), 669–683.
Huber, F., Koop, G., Onorante, L., Pfarrhofer, M., & Schreiner, J. (2020). Nowcasting in a pandemic using non-parametric mixed frequency VARs. *Journal of Econometrics*, http://dx.doi.org/10.1016/j.jeconom.2020.11.006, (forthcoming).
Jarocinski, M., & Lenza, M. (2018). An inflation-predicting measure of the output gap in the Euro area. *Journal of Money, Credit and Banking*, *50*(6), 1189–1224.
Joseph, A. (2019). Parametric inference with universal function approximators. arXiv preprint arXiv:1903.04209.
Kalli, M., & Griffin, J. E. (2014). Time-varying sparsity in dynamic regression models. *Journal of Econometrics*, *178*(2), 779–793.
Kastner, G., & Frühwirth-Schnatter, S. (2014). Ancillarity-sufficiency interweaving strategy (ASIS) for boosting MCMC estimation of stochastic volatility models. *Computational Statistics & Data Analysis*, *76*, 408–423.
Kayo, O. (2006). Locally linear embedding algorithm–extensions and applications. Manuscript.
Kelly, B. T., Pruitt, S., & Su, Y. (2019). Characteristics are covariances: A unified model of risk and return. *Journal of Financial Economics*, *134*(3), 501–524.
Koop, G., & Korobilis, D. (2012). Forecasting inflation using dynamic model averaging. *International Economic Review*, *53*(3), 867–886.
Koop, G., & Korobilis, D. (2013). Large time-varying parameter VARs. *Journal of Econometrics*, *177*(2), 185–198.
Koop, G., & Potter, S. M. (2007). Estimation and forecasting in models with multiple breaks. *Review of Economic Studies*, *74*(3), 763–789.
Korobilis, D. (2021). High-dimensional macroeconomic forecasting using message passing algorithms. *Journal of Business & Economic Statistics*, *39*(2), 493–504.
Lin, F., Yeh, C.-C., & Lee, M.-Y. (2011). The use of hybrid manifold learning and support vector machines in the prediction of business failure. *Knowledge-Based Systems*, *24*(1), 95–101.
McAdam, P., & McNelis, P. (2005). Forecasting inflation with thick models and neural networks. *Economic Modelling*, *22*(5), 848–867.
McAlinn, K., & West, M. (2019). Dynamic Bayesian predictive synthesis in time series forecasting. *Journal of Econometrics*, *210*(1), 155–169.
McCracken, M. W., & Ng, S. (2016). FRED-MD: A monthly database for macroeconomic research. *Journal of Business & Economic Statistics*, *34*(4), 574–589.
Medeiros, M. C., Vasconcelos, G. F., Veiga, Á., & Zilberman, E. (2021). Forecasting inflation in a data-rich environment: the benefits of machine learning methods. *Journal of Business & Economic Statistics*, *39*(1), 98–119.
Mullainathan, S., & Spiess, J. (2017). Machine learning: An applied econometric approach. *Journal of Economic Perspectives*, *31*(2), 87–106.
Oksanen, J., Blanchet, F. G., Friendly, M., Kindt, R., Legendre, P., McGlinn, D., Minchin, P. R., O'Hara, R. B., Simpson, G. L., Solymos, P., Stevens, M. H. H., Szoecs, E., & Wagner, H. (2019). Vegan: Community ecology package. R package version 2.5-6.
Orsenigo, C., & Vercellis, C. (2013). Linear versus nonlinear dimensionality reduction for banks' credit rating prediction. *Knowledge-Based Systems*, *47*, 14–22.
Pelger, M., & Xiong, R. (2021). State-varying factor models of large dimensions. *Journal of Business & Economic Statistics*, http://dx.doi.org/10.1080/07350015.2021.1927744, (forthcoming).
Pettenuzzo, D., & Ravazzolo, F. (2016). Optimal portfolio choice under decision-based model combinations. *Journal of Applied Econometrics*, *31*(7), 1312–1332.
Pfarrhofer, M. (2020). Forecasts with Bayesian vector autoregressions under real time conditions. arXiv preprint arXiv:2004.04984.
Raftery, A., Kárný, M., & Ettler, P. (2010). Online prediction under model uncertainty via dynamic model averaging: Application to a cold rolling mill. *Technometrics*, *52*(1), 52–66.
Ribeiro, B., Vieira, A., & das Neves, J. C. (2008). Supervised isomap with dissimilarity measures in embedding learning. In *Iberoamerican congress on pattern recognition* (pp. 389–396). Springer.
Richards, J., & Cannoodt, R. (2019). Diffusionmap: Diffusion map. R package version 1.2.0.
Richards, J. W., Freeman, P. E., Lee, A. B., & Schafer, C. M. (2009). Exploiting low-dimensional structure in astronomical spectra. *Astrophysical Journal*, *691*(1), 32–42.
Rossi, B., & Sekhposyan, T. (2019). Alternative tests for correct specification of conditional predictive densities. *Journal of Econometrics*, *208*(2), 638–657.
Roweis, S. T., & Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, *290*(5500), 2323–2326.
Saxe, A. M., Bansal, Y., Dapello, J., Advani, M., Kolchinsky, A., Tracey, B. D., & Cox, D. D. (2019). On the information bottleneck theory of deep learning. *Journal of Statistical Mechanics: Theory and Experiment*, *2019*(12), Article 124020.
Schölkopf, B., Smola, A., & Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, *10*(5), 1299–1319.
Stock, J., & Watson, M. (1999). Forecasting inflation. *Journal of Monetary Economics*, *44*(2), 293–335.
Stock, J. H., & Watson, M. W. (2002a). Forecasting using principal components from a large number of predictors. *Journal of the American Statistical Association*, *97*(460), 1167–1179.
Stock, J., & Watson, M. (2002b). Macroeconomic forecasting using diffusion indexes. *Journal of Business & Economic Statistics*, *20*(2), 147–162.
Stock, J. H., & Watson, M. W. (2007). Why has U.S. inflation become harder to forecast? *Journal of Money, Credit and Banking*, *39*(s1), 3–33.
Stock, J., & Watson, M. (2008). *Phillips curve inflation forecasts*: *NBER working papers 14322*, National Bureau of Economic Research, Inc.
Stock, J. H., & Watson, M. W. (2016). Core inflation and trend inflation. *The Review of Economics and Statistics*, *98*(4), 770–784.
Taylor, S. J. (1982). Financial returns modelled by the product of two stochastic processes-a study of the daily sugar prices 1961-75. *Time Series Analysis: Theory and Practice*, *1*, 203–226.
Tenenbaum, J. B., De Silva, V., & Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, *290*(5500), 2319–2323.
Timmermann, A. (2006). Forecast combinations. *Handbook of Economic Forecasting*, *1*, 135–196.
Zelnik-Manor, L., & Perona, P. (2004). Self-tuning spectral clustering. *Advances in Neural Information Processing Systems*, *17*, 1601–1608.
Zime, S. (2014). Economic performance evaluation and classification using hybrid manifold learning and support vector machine model. In *2014 11th International computer conference on wavelet actiev media technology and information processing* (pp. 184–191). IEEE.