Macroeconomic Forecasting with Large Language Models*

Andrea Carriero†, Davide Pettenuzzo‡, and Shubhranshu Shekhar§

March 19, 2025

Abstract

This paper presents a comparative analysis evaluating the accuracy of Large Language Models (LLMs) against traditional macro time series forecasting approaches. In recent times, LLMs have gained in popularity for forecasting due to their ability to capture intricate patterns in data and quickly adapt across very different domains. However, their effectiveness in forecasting macroeconomic time series data compared to conventional methods remains an area of interest. To address this, we conducted a rigorous evaluation of LLMs against traditional macro-prediction methods, using as common ground the FRED-MD database. Our findings provide valuable insights into the strengths and limitations of LLMs in the prediction of macroeconomic time series, shedding light on their applicability in real world scenarios.

Keywords: Large Language Models, Vector Autoregression, Factor Models, Forecasting

JEL classification: C11, C32, C51

1 Introduction

The recent emergence of Large Language Models (LLMs) has reshaped the landscape of natural language processing, ushering in a new era of computational linguistics. Bolstered by advancements in machine learning and deep neural networks, LLMs have garnered widespread attention for their remarkable ability to understand and generate human-like text. This transformative technology has revolutionized various applications, ranging from machine translation and sentiment analysis to chatbots and content generation. By leveraging vast amounts of text data and sophisticated algorithms, LLMs have demonstrated unparalleled proficiency in capturing linguistic nuances, contextual dependencies, and semantic meanings.[1] As a result, they are increasingly becoming invaluable tools for researchers, developers, and businesses.

*The views and opinions expressed here are those of the authors and do not necessarily reflect the views or positions of any entities they are affiliated with.

†School of Economics and Finance, Queen Mary University of London. Email: a.carriero@qmul.ac.uk

‡International Business School, Brandeis University. Email: dpettenu@brandeis.edu

§International Business School, Brandeis University. Email: sshekhar@brandeis.edu

[1]For example, the abstract of this paper has been generated with a single prompt by chatGPT 3.5.

In this paper we focus on an even more recent development, which is the use of LLMs to forecast time series data. While LLMs have predominantly been associated with natural language processing tasks, their versatility and adaptability have sparked interest in exploring their capabilities beyond linguistic domains. By leveraging the computational power and flexibility inherent in machine learning algorithms, LLMs promise to uncover intricate nonlinear relationships, capturing latent dynamics, and adapting to evolving data patterns.

This latest iteration of LLMs, trained specifically with the goal of forecasting time series data, is referred to as either Time Series Language Models (TSLMs) or Time Series Foundational Models (TSFMs) in the Machine Learning and Artificial Intelligence literature.[2] Several TSLMs have already been produced and are publicly available, including IBM's Tiny Time Mixers (Ekambaram et al., 2024), Time-LLM (Jin et al., 2024), LagLlama (Rasul et al., 2024), Google's TimesFM (Das et al., 2024) Salesforce's Moirai (Woo et al., 2024), and Nixtla's Time-GPT (Garza and Mergenthaler-Canseco, 2023). All these contributions show that TSLMs can produce gains in forecast accuracy, but in their empirical evaluations they tend to focus on a variety of datasets from very different domains and do not offer any specific information on how these models would fair in forecasting macroeconomic variables specifically.

Our first contribution in this paper is a thorough investigation on how TSLMs perform in predicting macroeconomic time series. A second contribution is a detailed comparison of their performance versus state-of-the-art time series methods such as Bayesian Vector Autoregressions (BVARs) and Factor Models as well as Bayesian Additive Regression Trees (BART), which has become a very popular nonparametric model for macroeconomic forecasting. In these endeavors, we focus on forecasting the variables contained in the FRED-MD dataset, a comprehensive repository of more than one hundred monthly macroeconomic variables curated by the Federal Reserve Economic Data (FRED) system.[3] This dataset encompasses a diverse array of economic indicators, and it is considered a benchmark in macroeconomic forecasting. A non-exhaustive lists of contributions using this data set to study the performance of alternative models in macroeconomic forecasting includes Stock and Watson (2006), Banbura et al. (2010), Carriero et al. (2016), Carriero et al. (2019), Korobilis and Pettenuzzo (2019), Koop et al. (2019), Chan (2022).

As noted in Faria-e Castro and Leibovici (2024), there are some characteristics of LLM models that makes it problematic to use them in the pseudo out-of-sample forecasting exercise that is typically performed in the empirical macroeconomics literature. One major challenge is that LLMs are pretrained

[2] In this paper, we prefer to use the TSLM acronym, to make it explicit that these models are derived from textual LLMs and as such share many similarities with them both in terms of architecture and model design.

[3] See https://research.stlouisfed.org/econ/mccracken/fred-databases/.

by developers on datasets of their choice, giving the researcher little to no control over the training data. For example, of the LLMs considered in this work, three out of five of them (including the best performing one, Moirai) list in their training data a large subset of the series we set out to forecast in this paper. To further complicate matters, since the training data lacks timestamps it's hard to retrain the model up to a certain date, making it difficult to reproduce the type of real-time analysis that an economic forecaster is mostly concerned with. Solving these issues by retraining LLMs on subsets of data and adding data to the training set is usually not possible due to high hardware requirements. But having said that, it is important to recognize that even including the dataset of interest in real time would not remove the issue of a contaminated training set, i.e. including information that the forecaster would not have access to in real-time.

Our empirical application is focused on forecasting the FRED monthly database using data ranging from 1960 to 2023. When working with traditional time series methods, we consider models of different cross-sectional dimensions, in order to understand the role that the information set has on the forecasting performance. The picture emerging from our analysis is one in which only two of the five TSLMs we consider are competitive against a simple AR benchmark (Salesforce's Moirai and Google's TimesFM). Moreover, when these more competitive TSLMs are stacked against Bayesian VARs and Factor Models, their forecasting performance is broadly comparable, if not slightly inferior. In particular, we find that the forecasting gains achieved by the econometric models tend to be more stable while TSLMs can perform very well for a handful of series but also show less reliability at times, as they seem to be more prone to generating the occasional unreasonable forecast.[4] On the other hand, TSLMs seem to work relatively better in the post-Covid-19 era (with the important caveat that the training set of these models does include information from the pandemic and post-pandemic period). While these results are based on using the TSLM models in a zero-shot forecasting manner, we also find that fine-tuning the TSLM models does not yield to significant improvements in their forecast accuracy.[5]

There are a few other papers that are looking at LLM in the macroeconomics and finance literature, and it is reasonable to expect that many more will follow. For example Bybee (2023) feed into an LLM model, OpenAI's GPT-3.5, a historical sample of news articles from The Wall Street Journal (WSJ) and ask the model to predict various financial and macroeconomic quantities, which they then aggregate into a time-

[4]This result may not appear to some as totally surprising. When working with textual LLMs, we are sometimes exposed to "Model hallucinations", i.e. the tendency of the model to generate content that is irrelevant, made-up, or inconsistent with the input data.

[5]By "zero-shot", we mean that the model is first trained on a large amount of data, and is then directly applied to solve a new task without re-training, or fine-tuning, its parameters on the new dataset.

series of monthly and quarterly expectations that are compared to a variety of existing survey forecasts. Chen et al. (2022) feed global news text data from Thomson Reuters Real-time News Feed (RTRS) and Third Party Archive (3PTY) in 13 different languages into pretrained LLMs, namely BERT (developed by Google), RoBERTa (by Meta), and OPT (by Meta), and use the resulting sentiment scores to predict firm-level daily returns. Kim et al. (2024) feed standardized and anonymous financial statements to GPT4 and instruct the model to analyze them to determine the direction of future earnings. They find that LLMs outperforms financial analysts in their ability to predict earnings changes. These approaches focus on leveraging the typical LLMs abilities in natural language processing, while this paper is focused on the recent efforts of training LLMs specifically for time-series forecasting. Chen et al. (2025) study whether both DeepSeek and ChatGPT can be used to predict the stock market and find that a good news ratio extracted from prompting ChatGPT is positively correlated with contemporaneous market returns and significantly predicts subsequent returns up to six months out. The closest work we are aware of in the macroeconomics literature is Faria-e Castro and Leibovici (2024), which looks at time series LLMs, but only focuses on one LLM and one target variable (inflation). In contrast to this, in this paper we take on the ambition goal of looking at the performance of a variety of LLMs, comparing their accuracy against state of the art macroeconomic forecasting approaches, across a large set of macroeconomic indicators.

The paper is organized as follows. Section 2 introduces LLMs and TSLMs and provides a discussion on how they are being leveraged to perform time series forecasting. Section 3 describes the econometric models we consider as state of the art benchmarks in our exercise. Section 4 describes the dataset and the design of the out-of-sample pseudo real time forecasting exercise and Section 5 presents our empirical results. Finally, Section 6 describes our fine-tuning process and the associated results, while Section 7 provides some concluding remarks and directions for further research. An Appendix provides technical details on the models considered and the data transformations used in the empirical analysis.

2 Foundational Models and LLMs

In recent years, Foundational Models (FMs), i.e. very large neural network models with billions of parameters have received tremendous attention in computer science fields due to their ability to capture complex relationships within massive datasets. One of the most astonishing features of FMs is their flexibility, i.e. their applicability to various tasks across many different domains in a zero-shot manner, effectively eliminating the need to train specialized models for each separate task. Within this context,

textual LLMs have emerged as a new way of understanding language, generating text and images, and conversing in natural language, and have completely revolutionized the way we interact with technology (Achiam et al., 2023). State-of-the-art textual LLMs and Chatbots, such as Open AI's ChatGPT and Meta's Llama, are now being used extensively across various natural language processing tasks, such as Information Retrieval and Scientific Research, but also Code Generation and Debugging.

2.1 Time Series Language Models (TSLMs)

Inspired by these successes, researchers are now exploring the usefulness of LLMs in other settings. Along these lines, Lu et al. (2022) find that LLMs trained exclusively on textual data can effectively solve sequence modeling tasks in other modalities, such as numerical computation, vision, and protein fold prediction. Among these extensions, we are witnessing the emergence of Time Series Language Models (TSLM), bridging the gap between LLMs original text data training and the numerical nature of time series data. These models are sometimes referred to in the literature as Time Series Foundational Models (TSFM). TSLMs are now being used to accomplish a variety of time series related tasks, ranging from prediction and classification to anomaly detection and data imputation. Notable examples of this new wave of model development include Rasul et al. (2024), Goswami et al. (2024), Ekambaram et al. (2024), Garza and Mergenthaler-Canseco (2023), Das et al. (2024), and Ansari et al. (2024). At the core of this very recent advancements is a simple but fundamental point. While there are in principle significant differences between a textual LLM, whose main task is essentially to predict the next word, and a time series forecasting model, whose main objective is to predict the next value of one or multiple time series jointly, inherently these two tasks are deeply intertwined. They both aim at learning the sequential structure of the data from the past and present history of some features and with that attempt to predict a sequence of future outcomes. In the case of a TSLM, the pretraining is done on a possibly very large set of time series, $\boldsymbol{X}_{1:T} = (x_{1,1:T}, ..., x_{N,1:T})$, and yields very large set of coefficients θ^* which describe a forecasting function f_{θ^*}. Then, given f_{θ^*} and the current and past values of a time series of interest y (not necessarily in the training set), the TSLM attempts to predict its future h values $y_{T+1:T+h}$ using[6]

$$\widehat{y}_{T+1:T+H|T} = f_{\theta^*}(y_{1:T}) = f(y_{1:T}|\boldsymbol{X}_{1:T}, \theta^*)\,. \tag{1}$$

[6]To keep the notation simple, we have assumed here a balanced set of features, i.e. all available over the same time period. If that is not the case, then we could instead write this as $\boldsymbol{X}_{1:T} = (x_{1,\underline{t}_1:T_1}, ..., x_{N,\underline{t}_N:T_N})$, where $\underline{t}_i$ and T_i would denote, respectively, the first and last observation of feature i, $i \in \{1, ..., N\}$.

The main difference is that while natural language consists of words from a finite vocabulary, time series are real-valued.

2.2 Components of Time Series Language Models

The earliest applications of LLMs to time series forecasting were built on simply prompting existing textual LLMs with time series data. For example, PromptCast (Xue and Salim, 2023) relies on converting numerical time series into text prompts and forecasting in a sentence-to-sentence manner.[7] There are multiple reasons why these approaches are sub-optimal. For once, these models requires a variable specific template to ask forecasting questions. In addition, using textual LLMs as is may lead to poor forecasting performances, mainly due to lack of real-world time series data in the training of these LLMs in the first place.

The most recent versions of TSLMs have evolved from this initial approach and now build on quantization-based methods, i.e. a process through which the numerical data is converted into discrete representations first, before becoming an input to the training of the LLM. We now turn to describing the basic building blocks of a TSLM, drawing whenever possible parallels to their textual counterparts for clarity. The specific choices and configurations of these building blocks define the unique characteristics of the various models. An overview of TSLM architecture is shown in Figure 1.

2.2.1 Tokenization

In textual LLMs e.g. ChatGPT, the process of text understanding begins with tokenization, which breaks down sentences into their fundamental semantic units – tokens. Tokens can be individual words, characters, or subwords. Tokens allow LLMs to identify and learn the relationships between semantic units, leading to a better text understanding. As we discussed above, time series data lack inherent well-defined semantic units such as words. To circumvent this issue, TSLMs rely on scaling, patching, and quantization techniques to transform numerical sequence data to a format "token" suitable for learning patterns and trends within data.[8] The common schemes utilized by current TSLMs are listed below. Note that not all TSLMs implement each scheme; they choose what works for their training process.

Patching

Patching is a process used to divide time series into shorter, fixed-length segments or windows called

[7] Here is a weather forecasting example, to demonstrate how this approach would work: *"From $t = 1$ to $t = T$, the average temperature of region i was X_i degree on each day. What is the temperature going to be on day $T + h$?"*

[8] It is worth noting here that a lack of fundamental semantic units presents also an opportunity to apply domain knowledge to derive better tokens, and thus potentially shepherding field of TSLMs for macroeconomic forecasting.

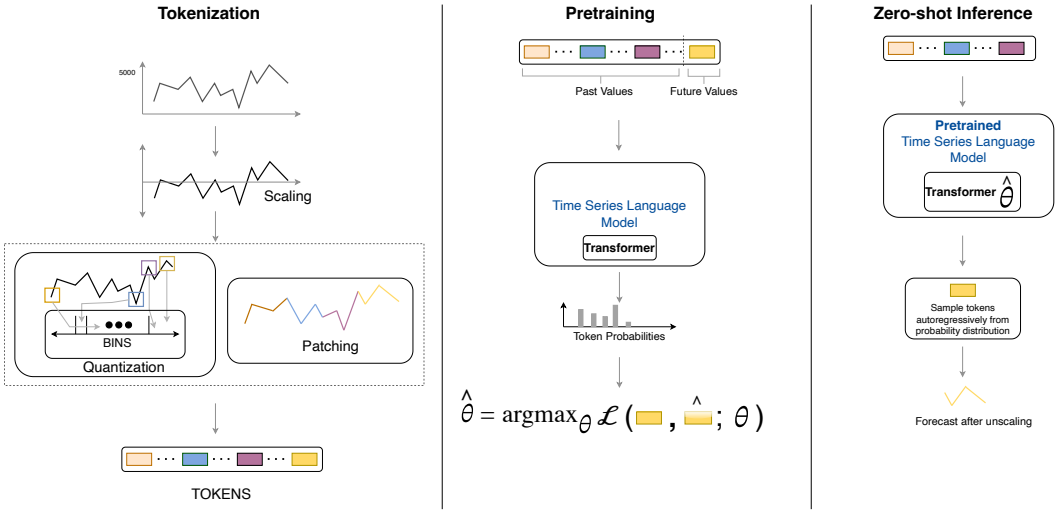


Figure 1: Components of a TSLM: The key components of a TSLM include tokenization, pretraining, and zero-shot forecasting. First, raw time series is tokenized into discrete tokens using techniques like scaling, quantization, and patching. Tokens are then used to pretrain a Transformer-based model that learns temporal dependencies and patterns. Finally, through zero-shot forecasting, the pretrained model can forecast in new and unseen time series domains. Note that different techniques at each stage–such as tokenization methods and model architectures–leads to different TSLM architectures.

patches. These patches can be treated as tokens, capturing local patterns. For example, Moirai, TimesFM, TTM use patching to create tokens, while LagLlama augments each patch with lag features and time-based features to construct tokens. Importantly, the patch size is itself a hyper-parameter that can be adjusted based on the specific characteristics of the time series data. For example, larger patch sizes are more suitable for high-frequency time series, while smaller patch sizes tend to be more suitable for low frequency time series. Consider as an example a time series $x_{1:T} = \{4.7, 4.76, 6.8, 7.2, 6.1\}$. Tokenization with patching size$=3$ and overlap$=2$, will yield following tokens $\{4.7, 4.76, 6.8\}$, $\{4.76, 6.8, 7.2\}$, $\{6.8, 7.2, 6.1\}$. Without overlap the tokens will be $\{4.7, 4.76, 6.8\}$, $\{7.2, 6.1\}$.

Quantization

Quantization is used to convert numerical time series values into a set of discrete tokens, similar to finite words in textual LLMs. This is achieved by dividing the value range of each time series, or a patch or set of patches (depending on the TSFM) into a predefined number of bins $\mathbb{B}$. Each data point is then assigned a token, a number between $\{1, \cdots \mathbb{B}\}$ based on the bin it falls into. There are different approaches to binning - uniform quantization divides the value range into bins of equal size, and data-dependent quantization adjusts the bin sizes based on the data distribution. CHRONOS (Ansari et al., 2024), for example, uses uniform quantization to encode time series into tokens. To continue with the previous example, let's apply

uniform quantization with $\mathbb{B} = 4$ to the time series $x_{1:T}$. The mapping from $x_{1:T}$ to the four bins is carried out as follows:

$$q_t = \begin{cases} 1 & \text{if } 4 \leq x_t < 5 \\ 2 & \text{if } 5 \leq x_t < 6 \\ 3 & \text{if } 6 \leq x_t < 7 \\ 4 & \text{if } 7 \leq x_t < 8 \end{cases}$$

Then the tokenized sequence is given by $q_{1:T} = \{1, 1, 3, 4, 3\}$.

Scaling[9]

When working with multiple time series at once, even when those are within the same dataset, it is common to have variables expressed with different scales. For example, in the case of macroeconomic variables we often see interest rates expressed in percentages, GDP growth reported in annualized growth rates, and unemployment rates displayed in absolute numbers. To ensure consistent processing, data points are typically scaled. Scaling also helps in optimization/learning for deep learning models, as varying scales may distort gradient computation. Scaling a value x_t can be done as follows,

$$\tilde{x}_t = \frac{x_t - M}{S}$$

where $\tilde{x}_t$ is the scaled value. Different choices for M and S lead to the types of scaling used by the various TSFMs. For example, LagLlama uses M = median of the context window and S = inter-quartile range within the context window.[10] It is also worth noting that scaling can be applied at the global level (using the value range of the entire time series), at a context window level, or at a patch level.

2.2.2 Model Architecture

Similar to textual LLMs, the transformer (Vaswani et al., 2017) is the core model architecture used in recent TSLMs.[11] Transformer-based models leverage self-attention mechanisms to capture long-range dependencies within time series, offering significant advantages in handling sequential data. Self-attention refers to the ability of transformers to focus on relevant parts of the time series data. In a

[9] Scaling can also be referred to as instance normalization or standardization.

[10] A context window is the length of a time series considered as history for forecasting.

[11] Additionally, there are non-tranformer-based architectures such as multi-layer perceptrons (MLP) used in some TSLMs. Of the various models we considered, all of them use a transformer architecture, except for TTM which uses a traditional MLP-based architecture.

transformer model, each token is projected into a higher dimensional space, allowing the model to capture richer information within that segment. Then, the self-attention layer applies linear transformations and uses the dot product between transformed vectors to identify relevant parts of the time series for each token. This helps in identifying long-range dependencies and relationships, agnostic of the token positions in the input time series. The original transformer architecture utilized two components, namely (i) an encoder, responsible for extracting the relevant information from the input time series and projecting/embedding it into a vector representation and (ii) a decoder, responsible for autoregressive prediction of next token conditioned on observed input sequence of tokens, and encoder output. Recently, several transformer architectures have evolved, notably encoder-decoder, decoder-only, and encoder-only. In the encoder-decoder setup (e.g. CHRONOS), the encoder maps an input sentence of some language to a continuous representation, and the decoder generates the translation token-by-token using the input representation and previously decoded tokens. In the decoder-only case (e.g. TimesFM, Das et al., 2024), the model only attends to tokens up to the current token, while in the encoder-only case (e.g. Moirai, Woo et al., 2024), the model uses information from the entire sequence to generate tokens. Note that encoder-decoder and decoder-only generate output tokens in an autoregressive manner in contrast to encoder-only architecture.

2.2.3 Time Series Augmentation

Textual LLMs train on massive text data readily available from books, articles, code, and web crawls. This abundance allows them to learn underlying patterns and relationships effectively. Similarly, TSLM models are intended for plug-and-play forecasting, which requires large-scale time series data with diverse patterns for training. However, time series data are inherently scarce. Augmentation helps mitigate the scarcity and variability of time series datasets by generating more diverse training data. There are several approaches to time series data augmentation (see Iglesias et al. (2023) for a detailed survey). These techniques typically involve convex combinations of existing sequences, combinations of ARMA processes, seasonal patterns, trends, and step functions, and combining the frequency spectrum of sequences and then converting it back to the time domain.

2.2.4 Pretraining and Fine-tuning

Pretraining

Pretraining refers to the process of feeding to the TSLMs a large amount of time series data so that the model can learn general temporal patterns and relationships and with that perform well across a wide spectrum of tasks. The time series data fed into the TSLM can be from various domains and span different frequencies and, as previously discussed, each time series will first be scaled and tokenized before any training occurs. Once the pretraining phase is completed, the TSLMs will have built a foundational knowledge base from a very general corpus of time series data, and with that can be used to perform zero-shot forecasting in a new domain.

Pretraining relies on self-supervised techniques that create tasks for the model so that it can learn directly from the input data without requiring human labeled examples. The success rate in these tasks can be measured in various ways, such as minimizing MSE or accurately predicting the quantized bins or the quantiles of the training data. As a concrete example, LagLlama pretraining looks for the model parameters that maximize the likelihood of the observed future values, under the assumption that these follow a Student-t distribution.

Fine-tuning

Fine-tuning is a training process that is used to adapt a pretrained TSLM, containing general foundational knowledge, to a specific task such as forecasting, or classification. Fine-tuning updates existing knowledge (parameters) of TSLM to better handle unique characteristics of the task-specific data. This can potentially improve the performance of a TSLM compared to zero-shot predictions for the given task. Fine-tuning can also be used to continually learn the model parameters and incorporate fresh data, improving TSLM's forecasting accuracy over time. For example, in macroeconomic forecasting, we can use fine-tuning to incorporate the data from the current quarter to update the parameters of the TSLM.

2.3 Relationship with Deep Learning

Before moving on to describe the existing TSLMs, it is worth briefly discussing the difference between Deep Learning (DL) models (see for example Torres et al. (2021), Lim and Zohren (2021) and Wen et al. (2022)) and TSLMs. DL models, share many similarities with traditional time-series methods, requiring

problem specific training data to learn model parameters and make forecasts. However, they differ the traditional methods in a number of ways – using non-linear architectures, large numbers of learnable parameters, and requiring much larger training data than traditional time series methods. TSLMs, on the other hand, are pretrained on large number of time series datasets. As with FMs, this pretraining allows them to perform "zero-shot" forecasting. This plug-and-play approach makes TSLMs particularly useful and powerful. Furthermore, pretraining allows TSLMs to require much less domain-specific knowledge data compared to deep learning models, and if needed can be fine-tuned on specific datasets.

2.4 State-of-the-art TSFMs

We now briefly describe the TSFM models considered in this paper.[12]

LagLlama (Rasul et al., 2024) LagLlama is a probabilistic foundation model that extends the Llama 2 architecture (Touvron et al., 2023) to time series data. LagLlama follows a z-score normalization at a window level, and tokenizes the input time series by extracting lagged features from the past values of the time series at different time lags. This allows the model to learn how past observations influence future ones. LagLlama is pretrained on 27 publicly available time series datasets from various domains. The model has about $\sim$ 2.5M parameters.

Moirai (Woo et al., 2024) Salesforce's Moirai is pretrained on 27B observations from time series datasets across nine domains. Moirai uses multiple patch (window) size projection layers to capture temporal patterns across various frequencies. It comes in 3 different model sizes – largest of which has about 311M parameters, which is significant in size compared to earlier models.

TTM (Ekambaram et al., 2024) IBM's TTM (Tiny Time Mixer), as the name suggests, is a small TSFM with $\leq$ 1M parameters that is pretrained on the Monash time series repository Godahewa et al. (2021) and LibCity Jiang et al. (2023) with $\sim$ 1B data points. TTM is based on TSMixer architecture (Ekambaram et al., 2023) which uses MLP-based (multi-layer perception) mixing across time steps and features. This enables multivariate forecasting through efficient extraction of temporal and cross-variate patterns for time series forecasting. TTM first normalizes each sequence to have zero mean and unit standard deviation. Then

[12] In addition to the models listed in this section, we have also experimented with MOMENT (Goswami et al., 2024), a family of time series transformer models that is pretrained on masked time series prediction task using patching. However this model is still in its very early stages and we have found its performance to be lagging significantly below the other TSFMs we have considered. We therefore did not include it in the comparison below.

it considers patches (windows) of varying length and varying resolution, since each dataset may have different optimal context to consider. TTM also applies downsampling on high resolution time series (e.g. seconds, minutes) to augment the training dataset.

Time-GPT (Garza and Mergenthaler-Canseco, 2023) Time-GPT was trained on a collection of publicly available time series, collectively encompassing over 100 billion data points. This training set incorporates time series from a broad array of domains, including finance, economics, demographics, healthcare, weather, IoT sensor data, energy, web traffic, sales, transport, and banking. Due to this diverse set of domains, the training dataset contains time series with a wide range of characteristics, ranging from multiple number of seasonalities, cycles of different lengths, and various types of trends.

TimesFM (Das et al., 2024) Google's TimesFM is a 200M parameter TSFM. It is trained on largest corpus of 100 billion time points that includes both real and synthetic time series. TimesFM breaks time series in patches and learns to predict subsequent patches, and uses a standard normalization at patch level.

Table 1 lists a few additional details on the various TSLMs described in this section. In particular, it provides information on when the model was released, i.e. when the pretraining sample ends, the list of all the data domains used in the pretraining phase of the model, and the size of training sample. It also flags whether each TSLM is univariate or multivariate.

3 Econometric Models

Previous work focusing on macroeconomic forecasting has evidenced that models which perform particularly well are Bayesian VARs (BVAR) and Factor Models. See e.g. Stock and Watson (2006), Banbura et al. (2010), Carriero et al. (2016), Carriero et al. (2019), Korobilis and Pettenuzzo (2019), Koop et al. (2019), Chan (2022). We therefore focus on these models, and in this section describe the specific choices we made in terms of their specifications. Starting with the BVARs, we consider both the popular natural conjugate Normal-Inverse Wishart prior, and an alternative specification which preserves conjugacy but allows for cross-shrinkage, a feature that has been shown to be particularly helpful in forecasting. We provide more details below.

Model	Release date	Training datasets (domains)	Size	Multivariate
LagLlama	Feb 2024	Traffic, Uber TLC, Electricity, London Smart Meters, Solar power, Wind farms, KDD Cup 2018, Sunspot, Beijing Air quality, Air Quality UC Irvine Repository, Huawei cloud, Econ/Fin*	352M tokens	✗
Moirai	Mar 2024	Energy, Transport, Climate, CloudOps, Web, Sales, Nature, Econ/Fin*, Healthcare	27B obs.	✓
TTM	April 2024	Electricity, Web traffic, Solar power, Wind farms, Energy consumption, KDD Cup 2018, Sunspot, Australian weather, US births, Bitcoin, Econ/Fin*	1B obs.	✓
Time-GPT	May 2024	Finance, economics, Demographics, Healthcare, Weather, IoT sensor data, Energy, Web traffic, Sales, Transport, and Banking	100B obs.	✗
TimesFM	May 2024	Google Trends, Wiki Page views, M4 Competion, Electricity and the Traffic data, Weather data, Synthetic Time Series Data	100B obs.	✗

Table 1: Training datasets for the various TSLMs considered in this paper. For each model, we provide a summary of when the model was pretrained, the list of all the datasets used in the pretraining phase of the model, the size of training sample, and whether it is multivariate. Additional details and references on the various datasets can be found in the individual TSLMs' technical papers. * indicates that the training data include the Monash forecast repository (available at https://forecastingdata.org), and therefore includes a large part of the FRED-MD dataset.

3.1 Bayesian VAR with Natural Conjugate Priors

Our starting point is a BVAR with the natural conjugate Normal-inverted Wishart (N-IW) prior. This prior dates back to Zellner (1971) and was later studied by Kadiyala and Karlsson (1997), Kadiyala and Karlsson (1993). In a seminal paper, Banbura et al. (2010) show that it can be successfully applied to a very large cross section of macroeconomic data. Several contributions followed that used this model and prior to handle large macroeconomic datasets.

Collect N different variables in the vector $y_t = (y_{1t}\ y_{2t}\ ...\ y_{Nt})'$, and write the Vector Autoregression or order p, $VAR(p)$, as:

$$y_t = \Phi_c + \Phi_1 y_{t-1} + \Phi_2 y_{t-2} + ... + \Phi_p y_{t-p} + \varepsilon_t;\ \varepsilon_t \sim i.i.d.N(0, \Sigma), \tag{2}$$

where $t = 1, \ldots, T$ denotes time. Note that with p lags and N variables, each equation has $K = Np + 1$ regressors. By grouping the coefficient matrices in the $N \times K$ matrix $\Phi' = [\Phi_c\ \Phi_1\ ...\ \Phi_p]$ and defining $x_t = (1\ y'_{t-1}\ ...\ y'_{t-p})'$ as a vector containing an intercept and p lags of y_t, the VAR can be written as:

$y_t = \Phi' x_t + \varepsilon_t$. Next, after stacking the $t = 1, \ldots, T$ equations by column and transposing we can rewrite the $VAR(p)$ using more compact matrix notation:

$$Y = X\Phi + E, \tag{3}$$

where $Y = [y_1, .., y_T]'$, $X = [x_1, .., x_T]'$, and $E = [\varepsilon_1, .., \varepsilon_T]'$ are $T \times N$, $T \times K$ and $T \times N$ matrices.

The conjugate N-IW prior elicits the prior on the coefficients conditionally on the error variance Σ:

$$\Phi|\Sigma \sim N(\Phi_0, \Sigma \otimes \Omega_0),\ \Sigma \sim IW(S_0, v_0). \tag{4}$$

Importantly, the Kronecker product implies that the error variance Σ enters the prior variance of the coefficients in a way that is symmetric and proportional to each equation. This in turn implies that the joint prior distribution of $p(\Phi\ , \Sigma\ )$ is a matricvariate t distribution. The conditional posterior distribution of this model is also N-IW (Zellner (1971)):

$$\Phi|\Sigma, Y \sim N(\bar{\Phi}, \Sigma \otimes \bar{\Omega}),\ \Sigma|Y \sim IW(\bar{S}, \bar{v}), \tag{5}$$

where $\bar{\Phi} = \bar{\Omega}(\Omega_0^{-1}\Phi_0 + X'Y)$, $\bar{\Omega} = (\Omega_0^{-1} + X'X)^{-1}$, $\bar{S} = S_0 + Y'Y + \Phi_0'\Omega_0^{-1}\Phi_0 - \bar{\Phi}'\bar{\Omega}^{-1}\bar{\Phi}$, and $\bar{v} = v_0 + T$. We elicit $\Phi_0, \Omega_0, S_0, v_0$ as in the Minnesota tradition, and more specific details on this can be found in Appendix A.1. To these priors we add the "sum of coefficients" and "single unit root" priors of Doan et al. (1984) and Sims (1993), which are both discussed also in Sims and Zha (1998). These priors were motivated by the need to avoid having an unreasonably large share of the sample period variation in the data accounted for by deterministic components. Again, more details can be found in Appendix A.1.

Two points worth discussing. First, all of these priors ultimately depend on a small set of hyperparemeters, which we estimate by maximizing the marginal likelihood of the model, and which is available in closed form. For more details see Appendix A.1. Second, note that the posterior variance of the coefficients has a Kronecker structure and therefore the joint prior distribution of $p(\Phi\ , \Sigma\ |Y)$ is matricvariate-t (and hence the prior is shown to be conjugate). The advantages of this setup are the fact that i) a closed form solution exists for the marginal likelihood, ii) since we know the form of the joint posterior distribution simulations can be performed via simple Monte Carlo sampling (as opposed to e.g. MCMC or SMC), and iii) the Kronecker structure ensures that the operations required to perform such sampling are at most of order $O(N^3)$. See Carriero et al. (2015) for a discussion. For these reasons, as

shown by Banbura et al. (2010), the BVAR with conjugate prior can handle very large datasets with ease.

3.2 Bayesian VAR with Asymmetric Conjugate Priors

One important limitation of the natural conjugate prior described above is that the Kronecker structure of the prior variance of the coefficients implies that the prior must be symmetric across equations. This in turn rules out cross-variable shrinkage, i.e. the possibility to shrink coefficients on lags of other variables more aggressively than those on own lags. For this reason, Chan (2022) developed an alternative specification that allows for asymmetry in the prior, while maintaining conjugacy. Rewrite the BVAR in (2) in its structural form, i.e.

$$Ay_t = b + B_1 y_{t-1} + B_2 y_{t-2} + \ldots + B_p y_{t-p} + u_t; \;\; u_t \sim i.i.d.\; N(0, D) \tag{6}$$

where A is a lower triangular matrix with ones on the diagonal and D is diagonal. This, in turns, allows for estimation of the model recursively, one equation at a time. It is easy to see that the structural parameters in (6) can be mapped to the reduced-form parameters in (2) by the following transformations: $\Phi_c = A^{-1}b$, $\Phi_j = A^{-1}B_j$ $(j = 1, ..., p)$, and $\Sigma = A^{-1}D\left(A^{-1}\right)'$.

Rewrite the i-th equation of the BVAR in (6) as

$$y_{i,t} = \widetilde{w}_{i,t}\alpha_i + \widetilde{x}_{i,t}\beta_i + u_{i,t}, \quad u_{it} \sim i.i.d.N(0, \sigma_i^2) \tag{7}$$

where $\widetilde{w}_{i,t} = (-y_{1,t}, \ldots, -y_{i-1,t})$, $\widetilde{x}_{i,t} = \left(1, y'_{t-1}, \ldots, y'_{t-p}\right)$, α_i includes the first $i-1$ elements on the i-th row of the matrix A, and β_i includes all the elements from the i-th of the matrix obtained after stacking together all the BVAR coefficients as follows: $\left(b, B_1, \ldots, B_p\right)$.

The asymmetric conjugate prior assumes that all the parameters of the BVAR are a priori independent across equations, $p\left(\alpha_1, \beta_1, \sigma_1^2, \ldots, \alpha_N, \beta_N, \sigma_N^2\right) = \prod_{i=1}^{N} p\left(\alpha_i, \beta_i, \sigma_i^2\right)$. The priors on the coefficients of each

equation i take the form:

$$\sigma_i^2 \sim IG\left(\frac{v_0 + i - N}{2}, \frac{s_i^2}{2}\right) \quad i = 1, ..., N \tag{8}$$

$$\alpha_{i,j} \big| \, \sigma_i^2 \sim N\left(0, \frac{\sigma_i^2}{s_j^2}\right) \quad 1 \leq j \leq i, i = 2, ..., N \tag{9}$$

$$\beta_i \big| \, \sigma_i^2 \sim N\left(m_i, \sigma_i^2 V_i\right) \quad i = 1, ..., N. \tag{10}$$

Chan (2022) shows that the prior on σ_i^2 and $\alpha_{i,j} \big| \, \sigma_i^2$ leads to a an Inverse Wishart prior for the reduced form error variance matrix, $\Sigma \sim IW(S_0, v_0)$, where $S_0 = \left(s_1^2, ..., s_N^2\right)$, and therefore is invariant with respect to the ordering of the variables. The degrees of freedom v_0, scale parameters s_i^2, and priors moments of the lagged coefficients (m_i and V_i) are elicited in line with the Minnesota tradition. In turn, these prior moments depend on a few hyperparameters set by maximizing the marginal likelihood, which also for this model is available in closed form. More details can be found in Appendix A.1.

3.3 Factor Model

Factor models are another class of models that has repeatedly shown to be particularly well suited for macroeocnomic forecasting. In this paper we use the implementation of McCracken and Ng (2016), which we now proceed to summarize. First, a set of static factors is estimated from the entire cross section of available data using Principal Components Analysis and the EM of Stock and Watson (2002) to balance the panel by filling in any missing values. Second, we use the extracted first factor to augment an autoregression of the i-th series with its lag values, i.e.:

$$y_{i,t} = \alpha_h + \beta_h(L)\hat{f}_{1t-h} + \gamma_h(L)y_{i,t-h} + \varepsilon_{i,t} \tag{11}$$

We estimate this model via OLS, and rely on BIC to select the number of lags of $\hat{f}_{1t}$ and y_{it}. The h-step ahead forecast is then computed as

$$\hat{y}_{i,t+h} = \hat{\alpha}_h + \hat{\beta}_h(L)\hat{f}_{1t} + \hat{\gamma}_h(L)y_{i,t} \tag{12}$$

Note that this approach (known as the "direct approach") is directly minimizing the h-step ahead forecast error, and it requires to perfrom a different regression for each forecast horizon.

3.4 Bayesian Additive Regression Trees

VARs and factor models assume linearity between endogenous variables and their lags or lags of their linear combinations. Although these models have proven quite successful at forecasting macroeconomic time series, the linearity assumption could be overly restrictive and this could have a negative impact on forecast accuracy. To test this hypothesis, we include Bayesian Additive Regression Trees (BART) in the pool of candidate models that we consider. BART (Chipman et al., 2010) models the conditional mean of the regression model by summing over a large number of trees which are, by themselves, constrained through a regularization prior. The main intuition behind the approach is that while each of these simple trees may only be able to explain a small fraction of the variation in the response variable, summing over a large number of them can describe the data extremely well. Prüser (2019), Huber and Rossini (2022), Huber et al. (2023) and Clark et al. (2023) are recent examples of successful applications of BART with macroeconomic data.

For each series i, we implement BART by estimating the following model[13]

$$y_{i,t} = \alpha_h + \gamma_h(L)y_{i,t-h} + f\left(X_{-i,t-h}\right) + \varepsilon_{i,t} \tag{13}$$

where $X_{-i,t-h}$ denotes all series at time $t-h$ other than $y_{i,t-h}$ and $g\left(X_{-i,t-h}\right)$ is a non-linear function obtained by summing over a larger number of regression trees,

$$f\left(X_{-i,t-h}\right) = \sum_{j=1}^{M} g\left(X_{-i,t-h}|\mathcal{T}_j, m_j\right) \tag{14}$$

where $g\left(X_{-i,t-h}|\mathcal{T}_j, m_j\right)$ identifies a single tree model with $\mathcal{T}_j$ denoting the tree structure and m_j is the vector of terminal node parameters associated with $\mathcal{T}_j$. Following, Chipman et al. (2010), we set $M = 250$ in our application, and specify a prior on the tree structure along the lines they suggested. The h-step ahead forecast is then computed as

$$\hat{y}_{i,t+h} = \hat{\alpha}_h + \hat{\gamma}_h y_{i,t} + \widehat{f}\left(X_{-i,t}\right) \tag{15}$$

As with factor models, this approach requires a different regression for each forecast horizon.

[13] We separate $y_{i,t-h}$ from the other predictors to avoid the model not recognizing that the lagged depedent variable is likely to have a non trivial role in predicting $y_{i,t}$. Our modeling choice is inspired by the approach of Carvalho et al. (2024), who modified the standard BART to introduce a Minnesota-type shrinkage specification into the node splitting selection. We also consider the alternative model $y_{i,t} = f\left(X_{t-h}\right) + \varepsilon_{i,t}$ but we found it to perform significantly below our preferred specification.

Abbreviation	Description	Transformation
PAYEMS	All Employees: Total nonfarm	5
INDPRO	IP Index	5
FEDFUNDS	Effective Federal Funds Rate	1
UNRATE	Civilian Unemployment Rate	1
RPI	Real personal income	5
DPCERA3M086SBEA	Real PCE	5
CMRMTSPLx	Real Manu. and TradeIndustries Sales	5
CUMFNS	Capacity Utilization: Manufacturing	1
CES0600000007	Avg Weekly Hours: Goods-Producing	1
HOUST	Housing Starts, Total	4
S&P 500	S&P's Common Stock Price Index: Composite	5
T1YFFM	1-Year Treasury C Minus FEDFUNDS	1
T10YFFM	10-Year Treasury C Minus FEDFUNDS	1
BAAFFM	Moodys Baa Corporate Bond Minus FEDFUNDS	1
EXUSUKx	U.S.-UK Foreign Exchange Rate	5
WPSFD49207	PPI: Final Demand: Finished Goods	5
PPICMM	PPI: Metals and metal products	5
PCEPI	Personal Consumption Expenditures	5
CES0600000008	Avg Hourly Earnings: Goods-Producing	6

Table 2: Variables in the Medium forecasting model. The column Transformation indicates the transformation of a series x_t, where: (1) no transformation, (2) Δx_t, (5) $\Delta \log(x_t)$, (6) $\Delta^2 \log(x_t)$ with Δ^i indicating i th differences.

4 Empirical Application

4.1 Data

We collect 120 monthly variables for the US spanning the period January 1959 to December 2023. The data, which are obtained from the Federal Reserve Economic Data (FRED) and are available at https://fred.stlouisfed.org, cover a wide range of key macroeconomic variables that applied economists monitor regularly, such as different measures of output, prices, interest and exchange rates, and stock market performance. We provide a full list of the data and their transformations in order to achieve stationarity in Appendix B. Out of the 120 series, we further distinguish a subset of "variables of interest", that is, key variables of interest which we will inspect very closely in order to evaluate how well the different models perform. These variables, along with their transformation codes, are described in Table 2.

4.2 Forecasting Exercise

We use the first twenty-five year of data, January 1960–December 1984, to obtain initial parameter estimates for all the econometric models, which are then used to predict outcomes from January 1985 ($h = 1$) to December 1985 ($h = 12$). The next period, we include data for January 1985 in the estimation sample, and use the resulting estimates to predict the outcomes from February 1985 to January 1986. We proceed recursively in this fashion until December 2022, thus generating a time series of forecasts for each forecast horizon h, with $h = 1, ..., 12$. Note that when $h > 1$, point forecasts are iterated and predictive simulation is used to produce the predictive densities.

Starting with the VARs, we estimate models of three different sizes: Medium (the 19 variables listed in Table 2), Large (the variables in medium plus an additional 20), and X-large (all 120 series available), that is, we consider 19, 39 and 120-variable sets. All models have a lag length of $p = 6$. For the factor models, we follow the same strategy. Namely, for each variable size and each sample, we first extract the first common factor, $\hat{f}_{1t}$. Next, for each variable in that same set and each forecast horizon, we run the factor-augment autoregression in (13) and forecast up to 12 months ahead. Similarly, for BART we re-estimate the trees for each variable set and sample, as the number of series in $X_{-i,t-h}$ changes with each variable set.

We proceed in a similar fashion with all TSLMs. More specifically, for each TSLM considered, starting with the first forecast date (December 1984), we feed the historical data to the pretrained model (when working with the univariate TSLMs we supplied one time series at a time, while with the multivariate TSLMs we supplied all time series within the specific model size at once). Next, we query the model to generate a zero-shot forecast for each of the variables provided, up to 12 steps ahead.[14] We also include a benchmark approach which uses OLS forecasts from univariate AR(1) models.

We evaluate the predictive accuracy of the various models, for each of the variables considered and each forecast horizon. In particular, we measure the precision of the h-step-ahead point forecasts for model i and variable j, relative to that from the univariate AR(1), by means of the ratio of Root MSFEs:

$$RMSFE_{ijh} = \sqrt{\frac{\sum_{\tau=\underline{t}}^{\bar{t}-h} e^2_{i,j,\tau+h}}{\sum_{\tau=\underline{t}}^{\bar{t}-h} e^2_{bcmk,j,\tau+h}}}, \tag{16}$$

where $\underline{t}$ and $\bar{t}$ denote the start and end of the out-of-sample period, and where $e^2_{i,j,\tau+h}$ and $e^2_{bcmk,j,\tau+h}$ are

[14] For all our experiments, we downloaded the open-sourced pretrained models and run them locally on our computer, using an Nvidia A6000 GPU with 48GB of memory. The only exception was Time-GPT, where we instead queried directly their API, with a separate query for each forecast date-time series combination.

the squared forecast errors of variable j at time τ and forecast horizon h associated with model i ($i \in \{\text{BVAR v1}, \text{BVAR v2}, \text{Factor model}, \text{BART}\} \cup TSLMs$) and the AR(1) model, respectively. The point forecasts used to compute the forecast errors are obtained by averaging over the draws from the various models' h-step-ahead predictive densities. Values of $RMSFE_{ijh}$ below one suggest that model i produces more accurate point forecasts than the AR(1) benchmark for variable j and forecast horizon h.

In closing this Section it is worth mentioning that in this paper we are only focusing on point forecast accuracy. This choice is intentional. While there is a rich literature on density forecasts for macroeconomic variables, using TSLM to construct density forecasts presents additional challenges, and we leave that to further research.

5 Results

5.1 Do some LLMs Perform Better than Others?

To begin, we focus on the overall forecasting performance of the various TSLMs under examination. Given that the evaluation sample we are considering include the Covid-19 period, to avoid conflating the overall picture with the large idiosyncrasies brought about by the pandemic, we focus first on the accuracy of the various models by stopping before the onset of the pandemic. That is, our evaluation sample stops in December 2019. Later on, we analyze how stable these are results are over time, and also look at the accuracy of the various model over the most recent period, January 2020 to December 2023. Figure 2 provides a synoptic view of the forecasting performance, as measured by RMSFE ratios relative to the AR benchmark. Each box-plot represents the interquartile range of the distribution of relative RMSFEs. When the mass is concentrated above the line corresponding to $RMSFE = 1$, this is an indication that the models do generally worse than the benchmark. The solid line within the box represents the median, while the whiskers represent the maximum and minimum data points that are not defined as outliers.[15]

Outliers are not depicted for scaling reasons, and instead we decided to separately report the maximum and minimum RMSFEs attained by each model in Table 3 (together with their median and standard deviation). Indeed, it does happen on several occasions that for some series the TSLMs produce forecasts that are patently unreasonable. These forecasts would be automatically disregarded by the user and would call for a fine-tuning of the models. This result highlights that while zero-shot TSLMs may

[15]Outliers are defined as values that are more than 1.5 times the interquantile range away from the top or bottom of the box (i.e. from the lower and upper quartiles).

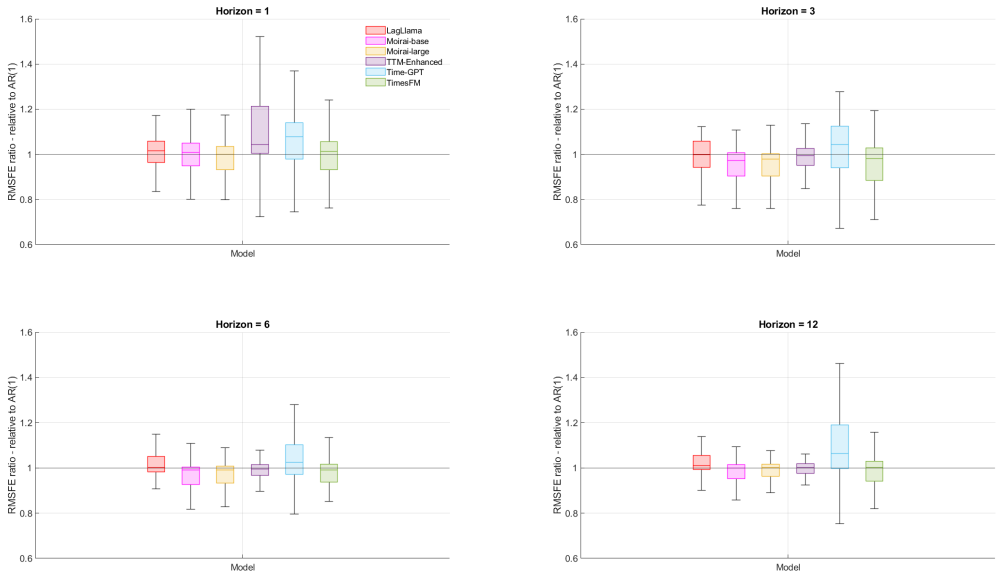


Figure 2: Distribution of RMSFEs (relative to an AR benckmark) for the TSLMs. The evaluation sample is January 1985 to December 2019.

produce on average reasonable forecasts, they should not be used mechanically and do require careful monitoring.

Having said that, the pattern emerging from a careful inspection of Figure 2 is one of heterogeneity among the various TSLMs. Specifically, some TSLMs show a performance that is consistently worse than others. These are TTM-enhanced and Time-GPT, for which the box plots are invariably positioned higher than those of the remaining TSLMs, signaling higher RMSFEs. These models also perform poorly compared to the benchmark, for example Time-GPT has large part of the mass of the box-plot above 1 for all horizons, and TTM for the one-step-ahead horizon. Lag-LLama performs a bit better than these two, however it is also systematically outperformed by the remaining models.

The TSLMs showing the best forecasting performance are Moirai and TimesFM. As we discussed, Moirai comes in three sizes, and the evidence shows that the largest size performs the best, with a mass of RMSFEs being consistently below 1 at all horizons. Because of these considerations, in the remainder of this Section, we will only focus on these best performing TSLMs (Moirai Large and TimesFM), and compare their performance against the BVARs and the factor model.

	h=1				h=3			
	Median	**Std**	**Min**	**Max**	**Median**	**Std**	**Min**	**Max**
LagLlama	1.015	1.057	0.726	7.271	0.997	0.787	0.633	4.843
Moirai-base	1.008	0.097	0.704	1.204	0.973	0.100	0.634	1.107
Moirai-large	0.999	0.102	0.703	1.436	0.978	0.099	0.637	1.158
TimesFM	1.014	0.129	0.706	1.482	0.980	0.127	0.635	1.318
TTM-Enhanced	1.044	0.352	0.723	2.959	0.993	0.108	0.718	1.448
Time-GPT	1.077	0.124	0.745	1.531	1.044	0.134	0.672	1.278
	h=6				**h=12**			
	Median	**Std**	**Min**	**Max**	**Median**	**Std**	**Min**	**Max**
LagLlama	1.002	0.461	0.568	3.577	1.009	0.260	0.597	2.431
Moirai-base	0.990	0.093	0.567	1.109	0.999	0.098	0.594	1.168
Moirai-large	0.991	0.096	0.600	1.159	1.001	0.113	0.619	1.324
TimesFM	0.990	0.142	0.593	1.629	1.001	0.158	0.482	1.440
TTM-Enhanced	0.995	0.097	0.643	1.400	1.002	0.198	0.663	2.257
Time-GPT	1.025	0.140	0.600	1.363	1.063	0.169	0.611	1.515

Table 3: Median, Std. deviation, Min and Max RMSFE by TSLM model type and forecast horizon. The evaluation sample is January 1985 to December 2019.

5.2 Differences in Point Forecast between LLMs and Econometric Models

Figure 3 provides a summary of the forecasting performance of the two best TSLMs contrasted with the performance of the econometric models under consideration. As before, the box-plots represent the interquartile range of the distribution of RMSFEs (relative to the AR(1) benchmark) and the whiskers represent the maximum and minimum data points that are not flagged as outliers.

Focusing first on the econometric models, it is remarkable that both the BVAR with natural conjugate prior and the factor model are consistently showing a good performance relative to the AR model. This is a well known result, which has been obtained by several different studies using this dataset. Surprisingly to us, the BVAR with asymmetric priors performs slightly worse than the symmetric counterpart, a result that is at odds with previous findings.

As already shown in Figure 2, Moirai Large and TimesFM also perform consistently better than the AR benchmark, but the distribution of RMSFEs is similar to that obtained with the econometric models. If anything, while all the models at almost all horizons have a median that is below the threshold of 1, there is a tendency for the distribution of the RMSFEs from TSLMs to have a more pronounced right tail, signaling an increased likelihood of observing large forecast errors. Instead, the econometric methods produce RMSFE distributions that are more skewed to the left, signaling a consistent better performance compared to the AR.

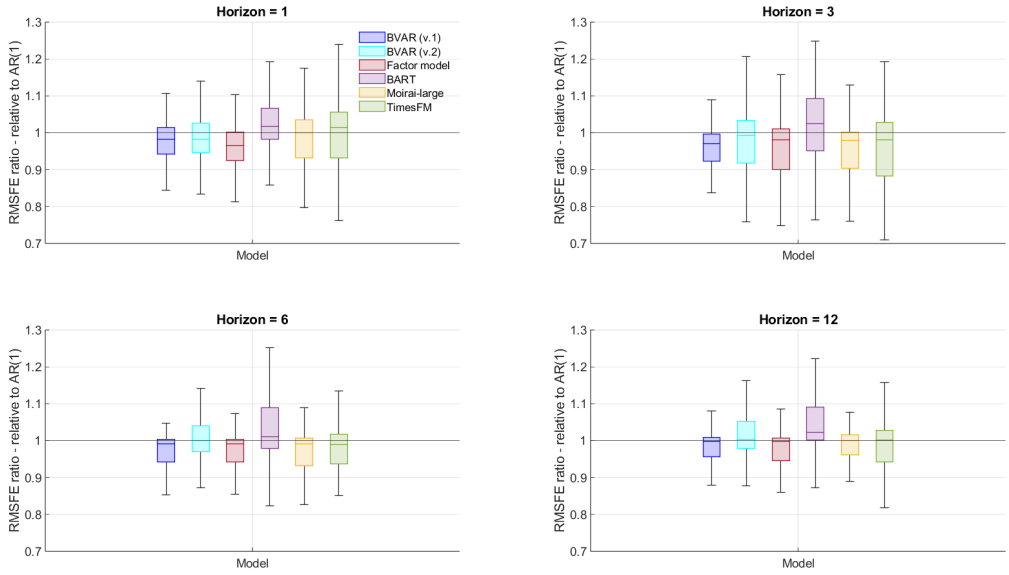


Figure 3: Distribution of RMSFEs (relative to an AR benckmark) for the econometric models and a selection of TSLMs. The evaluation sample is January 1985 to December 2019.

As before, additional descriptive statistics including the maximum and minimum relative RMSFEs (including outliers) are reported in Table 4. From this table it is clear that the econometric models do not present any extreme cases in which forecast errors are a multiple of those obtained by the AR benchmark, which it is instead the case for the TSLMs results shown in Table 3. Also, it is evident that the RMFSEs coming out of the econometric models are less dispersed. Overall the picture that emerges is one in which BVARs and factor models seem to offer a more stable and reliable performance, when compared to zero-shot TSLMs.

5.3 Are the Differences Statistically Significant?

The previous subsections have considered the distribution of RMSFEs relative to an AR benchmark where the distribution was computed across all of the series in the dataset. The fact that such distribution was to the left (right) of the threshold of 1 meant that overall a model was producing better (worse) forecasts than the AR model, but no assessment was made on whether each forecast individually was statistically different from the one produced by the benchmark. In this subsection we tackle this specific issue. To save space, we only consider the 19 variables included in the medium model. A full set of result, covering all

	h=1				h=3			
	Median	**Std**	**Min**	**Max**	**Median**	**Std**	**Min**	**Max**
BVAR (v.1)	0.983	0.060	0.827	1.158	0.970	0.060	0.767	1.089
BVAR (v.2)	0.982	0.081	0.800	1.315	0.992	0.099	0.732	1.243
Factor model	0.965	0.065	0.766	1.103	0.980	0.102	0.682	1.183
BART	1.017	0.093	0.858	1.420	1.024	0.132	0.763	1.381
	h=6				**h=12**			
	Median	**Std**	**Min**	**Max**	**Median**	**Std**	**Min**	**Max**
BVAR (v.1)	0.991	0.068	0.725	1.106	0.999	0.080	0.671	1.162
BVAR (v.2)	1.000	0.095	0.714	1.222	1.004	0.111	0.611	1.343
Factor model	0.992	0.087	0.602	1.105	0.999	0.089	0.633	1.125
BART	1.011	0.131	0.757	1.457	1.023	0.154	0.670	1.905

Table 4: Median, Std. deviation, Min and Max RMSFE by model type and forecast horizon. The evaluation sample is January 1985 to December 2019.

120 variables, is available upon request.

To provide a rough gauge of whether the RMSE ratios are significantly different from 1, we use the Diebold and Mariano (1995) t-statistic (DM). Our use of the DM test with models that are in some case nested is a deliberate choice. Monte Carlo evidence in Clark and McCracken (2012, 2015) indicates that, with nested models, the Diebold-Mariano test compared against normal critical values can be viewed as a somewhat conservative (conservative in the sense of tending to have size modestly below nominal size) test for equal accuracy in the finite sample. Differences that are statistically significant are denoted by one, two, or three stars, corresponding to significance levels of 10%, 5%, and 1%, respectively. The underlying p-values are based on t-statistics computed with a serial correlation-robust variance, using a rectangular kernel, $h-1$ lags, and the small-sample adjustment of Harvey et al. (1997).

Results for forecast horizons of one and three steps ahead are displayed in Table 5. At these short forecast horizons BVARs and factor models outperform the AR benchmark in many cases (12, 13, and 14 cases respectively for BVAR(v.1), BVAR(v.2), and Factor model). In these instances, the forecasting gains are both large and statistically significant. In the remaining few cases in which the econometric models do not outperform the benchmark, the forecasting losses are small and never statistically significant. The TSLMs outperform the AR in fewer instances (5 for Moirai and 6 for TimesFM), also with gains that are both large and statistically significant. In the remaining cases, the forecasting losses are not statistically significant. As we have seen before, there is a tendency for TSLMs to produce the occasional more extreme results, with larger losses than those observed for the econometrics models.

Results for forecast horizons of 6 and 12 step-ahead are displayed in Table 6. At these horizons, for

	$h = 1$											
	BVAR(v.1)		BVAR(v.2)		Factor model		BART		Moirai Large		TimesFM	
PAYEMS	0.84	***	0.83	***	0.87	***	0.97		0.83	***	0.82	***
INDPRO	0.90	***	0.93	**	0.97		1.02		0.95	**	0.94	**
FEDFUNDS	1.21		0.97		0.87	**	1.34		1.00		1.23	
UNRATE	0.88	***	0.84	***	0.89	***	0.94		1.04		1.11	
RPI	0.98		0.98		0.99		1.05		1.00		1.04	
DPCERA3M086SBEA	1.00		0.99		0.98	*	1.03		1.02		1.06	
CMRMTSPLx	0.96	*	0.95	**	0.98		1.01		1.01		1.04	
CUMFNS	0.90	*	0.91	**	0.95		0.96		1.09		1.48	
CES0600000007	0.90	***	0.91	***	0.91	**	1.05		0.93	*	0.94	**
HOUST	0.95	***	0.95	**	0.93	***	1.01		1.00		0.92	***
S&P 500	1.04		1.03		1.00		1.03		1.03		1.02	
T1YFFM	1.19		1.18		1.06		1.28		1.04		1.13	
T10YFFM	1.06		1.00		1.00		1.09		1.04		1.16	
BAAFFM	1.02		0.96		0.94		1.07		1.01		1.28	
EXUSUKx	1.02		1.03		1.01		1.05		1.02		1.01	
WPSFD49207	0.97		1.00		1.03		1.03		1.01		1.01	
PPICMM	1.00		1.01		0.99		1.02		1.03		1.04	
PCEPI	0.94	***	0.94	**	0.98		1.02		0.99		0.95	*
CES0600000008	0.85	***	0.85	***	0.78	***	0.91	***	0.77	***	0.78	***
	$h = 3$											
	BVAR(v.1)		BVAR(v.2)		Factor model		BART		Moirai Large		TimesFM	
PAYEMS	0.76	**	0.78	***	0.79	**	0.99		0.77	***	0.75	**
INDPRO	0.94		0.98		0.99		0.98		0.97		0.94	
FEDFUNDS	1.24		1.04		0.96		1.09		0.91		1.16	
UNRATE	0.74	**	0.72	***	0.73	***	0.98		0.91		0.78	**
RPI	1.00		1.00		0.99		1.02		1.00		1.01	
DPCERA3M086SBEA	1.00		1.00		1.00		1.01		0.99		1.00	
CMRMTSPLx	0.97	**	0.98		0.98		1.01		1.00		1.01	
CUMFNS	0.79	**	0.85	**	0.92		1.07		0.93		1.32	
CES0600000007	0.72	***	0.79	***	0.79	***	0.80	***	0.79	***	0.83	***
HOUST	0.99		1.02		0.93	**	1.04		1.01		0.78	***
S&P 500	1.02		1.02		1.01		1.03		1.00		1.03	
T1YFFM	1.32		1.30		1.14		1.32		1.03		1.12	
T10YFFM	1.13		1.08		1.04		1.05		0.96		1.13	
BAAFFM	1.09		1.05		1.01		1.05		0.93		1.17	
EXUSUKx	1.03		1.05		1.01		1.06		1.02		1.02	
WPSFD49207	0.99		1.02		1.05		1.06		0.99		1.03	
PPICMM	1.00		1.02		1.00		1.05		1.00		1.03	
PCEPI	0.93	**	0.95		0.97		0.98		0.89	***	0.91	***
CES0600000008	0.83	***	0.83	***	0.77	***	0.91	***	0.74	***	0.75	***

Table 5: Forecast horizons 1 and 3 step-ahead. RMSFE relative to the AR(1). Differences in accuracy that are statistically significant at 10%, 5%, and 1% levels are denoted by one, two, or three stars, respectively. The evaluation sample is January 1985 to December 2019.

	$h = 6$											
	BVAR(v.1)		BVAR(v.2)		Factor model		BART		Moirai Large		TimesFM	
PAYEMS	0.99	*	0.99	*	0.99		0.81	**	0.99	**	0.99	*
INDPRO	0.99		1.00		1.01		1.01		0.99		0.98	
FEDFUNDS	1.08		1.04		0.95		1.36		0.93		1.21	
UNRATE	0.92	**	0.92	***	0.90	**	0.85		0.95		0.94	**
RPI	1.00		1.00		1.00		1.03		1.00		1.01	
DPCERA3M086SBEA	1.00		1.00		1.00		1.01		1.00		1.00	
CMRMTSPLx	1.00		1.00		1.00		0.98		1.01		1.02	
CUMFNS	0.87	*	0.90	*	1.01		1.01		0.92		1.14	
CES0600000007	0.69	***	0.76	***	0.78	***	0.87	***	0.76	***	0.79	***
HOUST	1.01		1.05		0.94	*	0.99		0.98		0.66	**
S&P 500	1.01		1.01		1.00		1.05		1.00		1.03	
T1YFFM	1.29		1.35		1.02		1.33		1.07		1.20	
T10YFFM	0.98		1.05		0.92	***	1.09		0.97		1.14	
BAAFFM	1.03		1.06		0.98		1.07		0.93		1.15	
EXUSUKx	1.02		1.04		1.00		1.09		1.00		1.02	
WPSFD49207	1.01		1.04		1.03		1.07		0.99		1.02	
PPICMM	1.02		1.04		1.00		1.04		1.01		1.06	
PCEPI	0.92		0.96		0.92		1.00		0.85	***	0.85	***
CES0600000008	0.85	***	0.84	***	0.78	***	0.83	***	0.76	***	0.76	***
	$h = 12$											
	BVAR(v.1)		BVAR(v.2)		Factor model		BART		Moirai Large		TimesFM	
PAYEMS	1.00		0.99		1.00		1.00		1.00		0.99	
INDPRO	1.01		1.01		1.01		1.04		1.01		0.99	
FEDFUNDS	0.96		1.01		0.96		0.86		1.00		1.08	
UNRATE	0.89	***	0.88	***	0.86	**	0.99		0.96		0.89	*
RPI	1.00		1.00		1.00		1.01		1.00		1.00	
DPCERA3M086SBEA	1.00		1.00		1.00		1.01		1.00		1.00	
CMRMTSPLx	1.01		1.01		1.01		1.03		1.02		1.02	
CUMFNS	0.91		0.93		1.06		1.08		0.92		1.02	
CES0600000007	0.70	***	0.75	***	0.81	***	0.80	***	0.76	***	0.78	***
HOUST	1.03		1.11		1.02		1.18		0.96		0.48	**
S&P 500	1.01		1.01		1.00		1.04		1.01		1.05	
T1YFFM	1.25		1.50		1.01		1.33		1.11		1.25	
T10YFFM	0.87		1.05		0.90	**	0.88	**	1.03		1.15	
BAAFFM	0.94		1.01		0.96		0.82		0.96		1.22	
EXUSUKx	1.00		1.01		1.00		1.09		1.03		1.00	
WPSFD49207	1.02		1.05		1.03		1.05		1.00		1.03	
PPICMM	1.00		1.00		1.01		1.04		1.01		1.03	
PCEPI	0.90	*	0.97		0.90	**	0.94		0.82	***	0.83	***
CES0600000008	0.88	***	0.88	***	0.81	***	0.91	**	0.78	***	0.80	***

Table 6: Forecast horizons 6 and 12 step-ahead. RMSFE relative to the AR(1). Differences in accuracy that are statistically significant at 10%, 5%, and 1% levels are denoted by one, two, or three stars, respectively. The evaluation sample is January 1985 to December 2019.

the 19 series in this medium dataset, improvements versus the benchmark are less clear-cut. In general, neither the econometric models nor the TSLMs systematically outperform the AR benchmark, even though it is worth noting that there are no instances in which they significantly under perform. Overall, Moirai is the model that more often produced better forecasts (in 13 cases at the 6-step ahead and 10 cases at the 12-step ahead) followed closely by the BVAR(v1) and factor model. For some series, there are significant gains. In particular the TSLMs produce good forecasts for Personal Consumption Expenditures (PCEPI) and Hourly Earnings (CES0600000008), with gains that are significant and larger than those obtained by the econometric models. The econometric models produce good forecasts of Unemployment rate (UNRATE) and Weekly hours (CES0600000007).

To summarize the empirical results discussed so far: i) both the econometric and the TSLM models are competitive against the AR benchmark, and never significantly outperformed by it, ii) at short horizons, the econometric models outperform the AR benchmark, while at longer horizons there is a 50:50 split between gains and losses iii) TSLMs do very well for a handful of series across all horizons.

5.4 Results during the Covid-19 Pandemic

The results presented so far gave an overall picture of the accuracy of the various models across the different forecast horizons, focusing on the evaluation sample January 1985 to December 2019. In this section, we focus on the post Covid-19 period sample, Jan 2020 to December 2022 (to reduce the impact on the results of the extreme outliers right around the onset of the pandemic, we excluded from our analysis the forecasts made between March and June 2020).[16]

Figure 4 and Figure 5 replicate our pre-Covid-19 analysis, again first comparing the forecast accuracy of the various TSLMs, and next stacking that against the BVARs and factor model. Compared to what we shown in Figure 2 and Figure 3 there is an overall deterioration in forecast accuracy for all the models, which all struggle in overperforming the benchmark. Within the LLMs, only Moirai-large stands out, with a performance comparable with that of the factor model and the BVAR with symmetric conjugate priors. It's worth pointing out that in this sample the econometric models tend to make at times large forecast errors, mainly during the second half of 2020. In the case of the factor model it happens especially for interest rates and spreads.

[16]The choice of excluding from the sample the most problematic Covid-19 related observation is based exclusively on simplicity. Still, there is evidence that simple approaches such as this might work well, see Schorfheide and Song (2021). One could also consider more sophisticated ways to deal with the Covid-19 observations such as Carriero et al. (2022) or Lenza and Primiceri (2022).

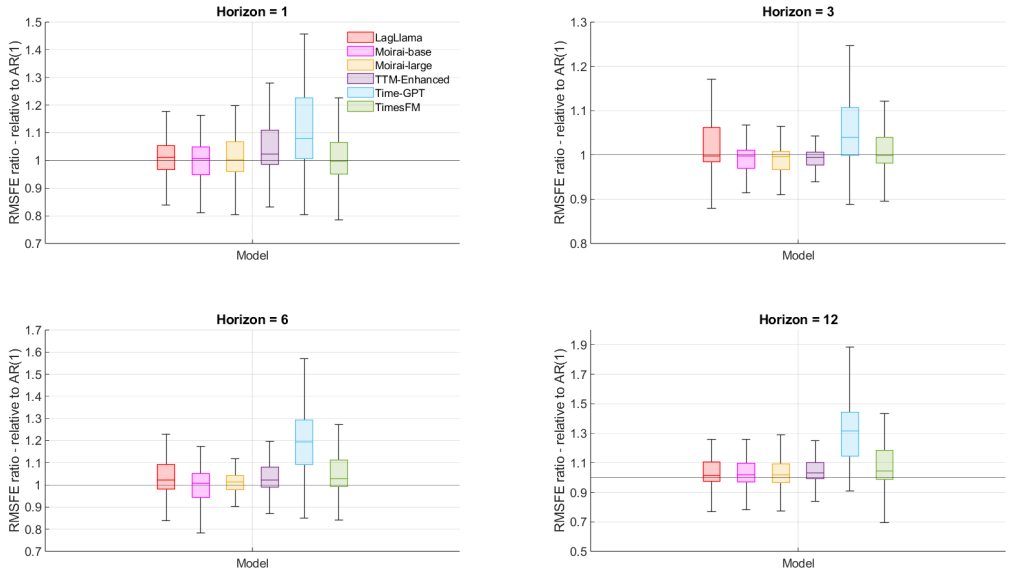


Figure 4: Distribution of RMSFEs (relative to an AR benckmark) for the TSLMs. The evaluation sample is January 2020 to December 2022 (excluding the months from March to June 2020).

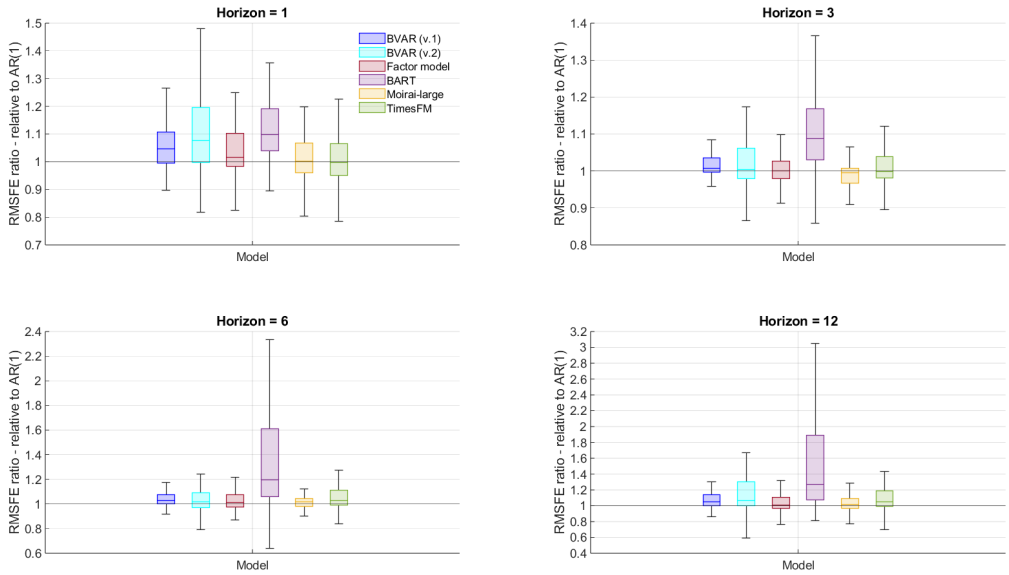


Figure 5: Distribution of RMSFEs (relative to an AR benckmark) for the econometric models and a selection of TSLMs. The evaluation sample is January 2020 to December 2022 (excluding the months from March to June 2020).

The usual caveat about TSLMs should be kept in mind, i.e. the fact that, as shown in Table 1, these models do include the pandemic period in their training sample (and a few of these models also include may of the macroeconomic variables we set out to forecasts), and this makers it harder to pin down to which extent this forecasting performance could have been achieved in real time.

5.5 How do TSLMs Work with Persistent Series?

As described above, some of the series are transformed before entering the estimation stage of the models. Looking at the results shown so far, there appears to be some indication that TSLMs perform relatively better when the degree of persistence of the series they utilize is relatively low. To further investigate this aspect, we break down the previous results by separately looking at series with different degrees of persistence. Specifically, Figure 6 shows results for variables for which the one lag partial autocorrelation is below 0.9, while Figure 7 shows results for variables for which the one lag partial autocorrelation is above 0.9. The latter variables are of course highly persistent.

For the econometric models, the pattern emerging from this particular cut of the results is in line with expectations. Specifically, VARs perform relatively better when estimated with more persistent variables (likely because keeping the variables in levels allows for common trends and co-integration) while factor models perform relatively better when estimated with less persistent variables (likely because the estimation of the factors relies on stationarity). This is evident in the graphs, where the factor model box-plots are consistently below 1 in Figure 6 while the BVAR box-plots are consistently below 1 in Figure 7.

Turning the attention to TSLMs, for series with low to moderate persistence, they seem to perform relatively better, even though it should be noted that they are still outperformed by factor models for this subset of series. The overall picture changes starkly when considering highly persistent variables. In these cases, one can observe a definite increase in the variation of the performance of TSLM.

To give an example of the extent of variation in performance, Figure 8 shows the time series of out-of-sample forecasts produced by the factor model, the BVAR (with symmetric priors) and TimesFM model for a few selected variables. While for the 1-step ahead horizons the forecasts are quite similar (left side panels), large differences emerge at the 12-step ahead horizon (right side panels).

For Housing Starts (HOUST), the 12-step ahead forecasts produced by TimesFM are remarkably good, as this model is able to almost perfectly predict the pronounced slowdown that happened during the

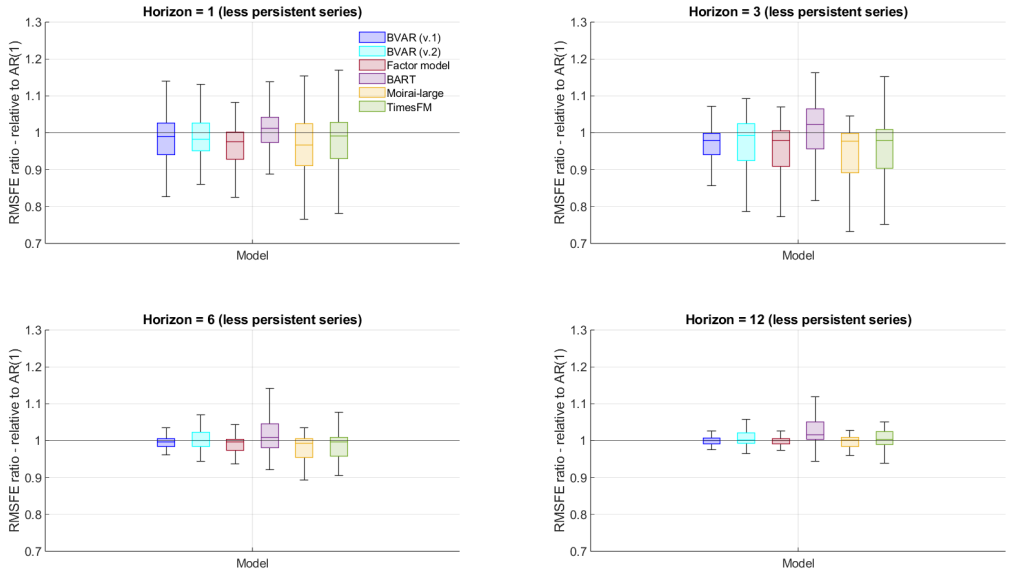


Figure 6: Distribution of RMSFEs (relative to an AR benckmark) for the econometric models and a selection of TSLMs, focusing on variables with medium and low persistence. The evaluation sample is January 1985 to December 2019.

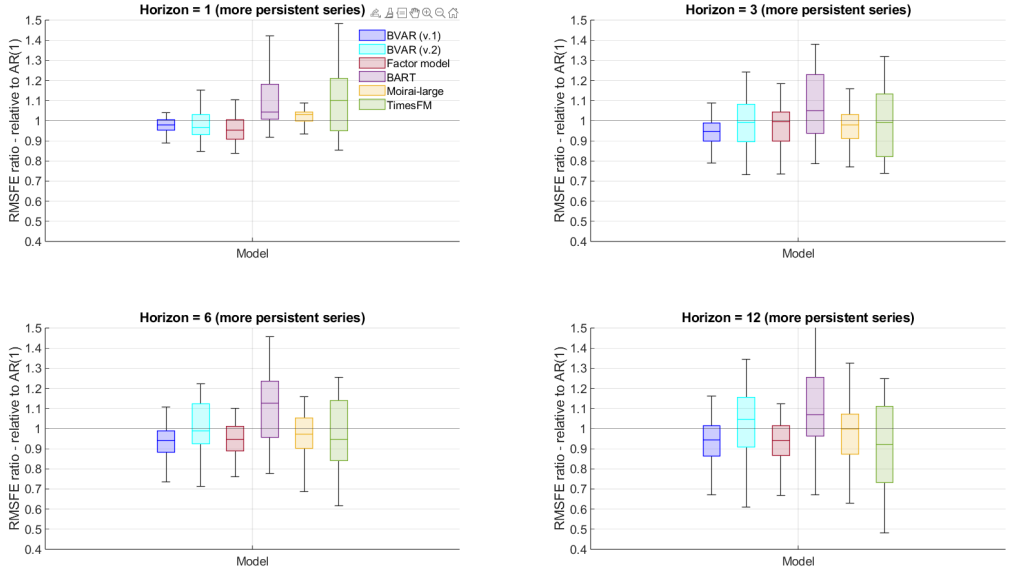


Figure 7: Distribution of RMSFEs (relative to an AR benckmark) for the econometric models and a selection of TSLMs, focusing on variables with high persistence. The evaluation sample is January 1985 to December 2019.

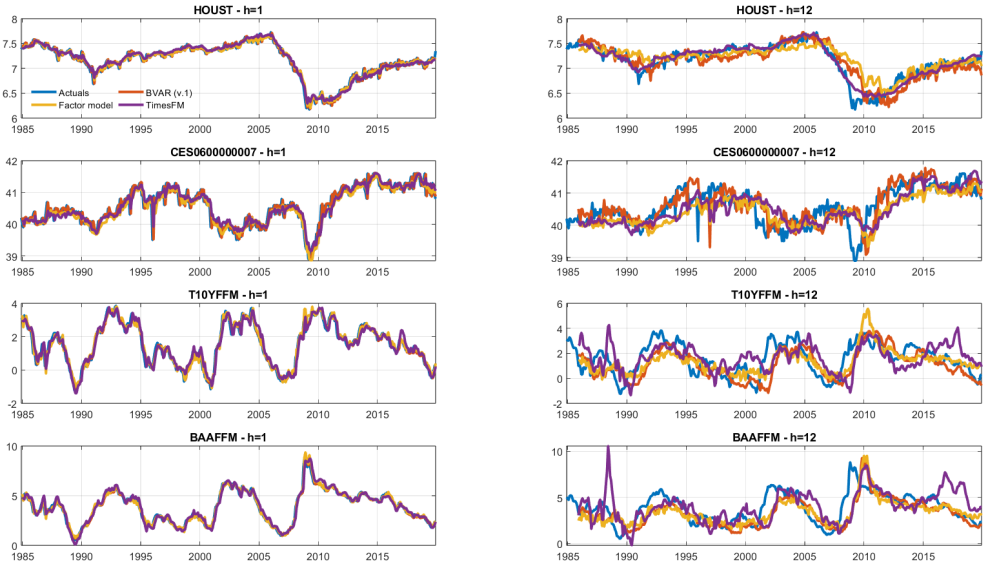


Figure 8: Time series of out-of-sample forecasts produced by the factor model, the BVAR with symmetric priors and the timesFM model, for selected variables. Left panels show 1-step ahead forecasts, and right panels show 12-step ahead forecasts. The sample is January 1985 to December 2019.

2007 financial crisis and the ensuing recession. This results in an astonishing 50 percent improvement in RMSFE compared to the AR benchmark. Instead, the econometric models only pick up this slowdown with a delay, and the factor model even fails to recognize the depth of the trough. Of course, one must keep in mind that TimesFM includes this recessionary time period in its training sample, so it seems to be on some level aware that a change in time series behavior is to be expected within these dates. For weekly hours (CES0600000007) TimesFM outperforms the factor model but underperforms the BVAR. For variables such as the spread between the 10 year interest rate and the Federal Funds rate (T10YFFM) and the spread between corporate bonds and the Federal Funds rate (BAAFFM), the performance of TSLMs is subpar, featuring entire periods of "hallucination" in the second half of the 80s and in 2017-2019, resulting in RMSFE losses of the order of 15-20 percent.

6 Fine-tuning

Up until now, all TSLMs results have been based on zero-shot predictions of the original pretrained models. As discussed in subsection 2.2, alternatively one could consider the option of fine-tuning the

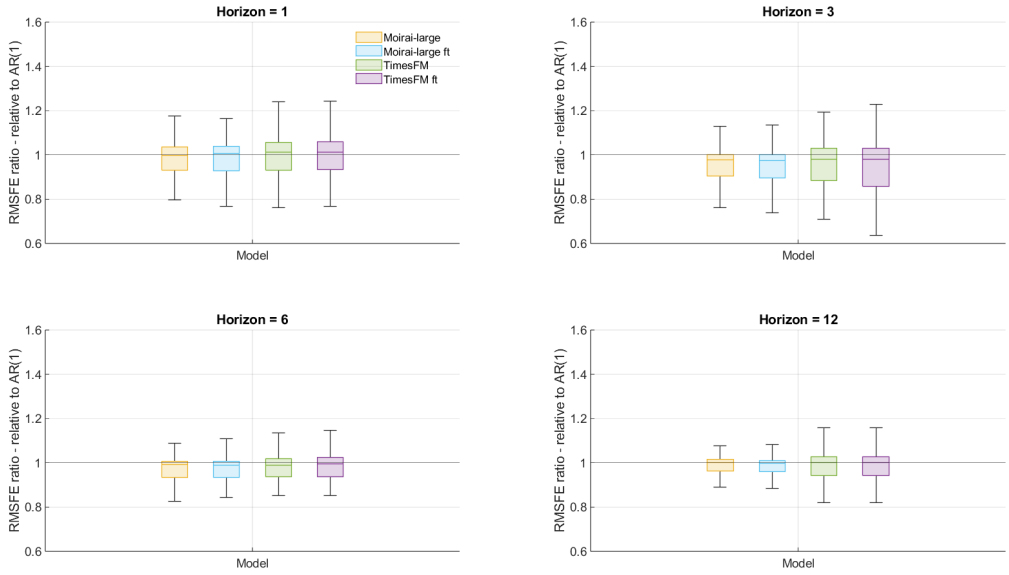


Figure 9: Distribution of RMSFEs (relative to an AR benckmark) for TimesFM and Moirai-large for zero shot and fine-tuning. The evaluation sample is January 1985 to December 2019.

TSLMs, i.e. updating the pretrained model's parameters on a task-specific dataset, with the goal of improving forecasting performance. Broadly speaking, to implement fine-tuning one could follow one of two approaches, (i) Continued-pretraining or (ii) Parameter-efficient fine-tuning.[17] Continued-pretraining involves further training the model on a domain-specific dataset starting with pretrained parameters, and allowing all model parameters to adapt to the characteristics of the given data. This can lead to overfitting when the fine-tuning data is small. Parameter-efficient methods instead freeze some of the original model's parameters and introduce trainable low-rank matrices (e.g. low-rank adaptation (LoRA) methods as in (Hu et al., 2022)), significantly reducing the number of trainable parameters, addressing overfitting, and requiring comparatively less computational resources.

Here, we selected the two best models from our zero-shot experiments – TimesFM and Moirai-large – and fine-tuned them on FRED-MD data. Note that some of the pretrained TSLMs, including Moirai, are trained on datasets that overlap with the FRED-MD. However, the influence of any single dataset in the pretraining phase gets diluted within the vast and diverse pretraining corpus. Consequently, the models learn general time series patterns, rather than specializing in the specific characteristics of FRED-MD. To make a TSLM more dataset-specific, fine-tuning is required. The main challenge we face in this context is

[17] See Han et al. (2024) for a detailed review.

that FRED-MD is a relatively small dataset, which may lead to overfitting problems.[18] To address these challenges, we employ a simple parameter-efficient fine-tuning strategy as opposed to continued-pretraining. In particular, we freeze the transformer layers of the two TSLMs, thereby reducing the number of trainable parameters.[19] Note that freezing the pretrained model's weights (parameters) is a special case of LoRA, which additionally injects trainable rank decomposition matrices into each layer of TSLM. After freezing the transformer layers, the number of trainable parameters reduced to approximately $\leq$ 10M for both TimesFM (instead of 200M) and Moirai-large (instead of 311M), primarily part of the final layers.[20] Throughout, to keep computational costs manageable, we chose to fine-tune each model once every 12 months, and the resulting models were then used for zero-shot forecasting over the subsequent 12 months.[21] We run the fine-tuning experiment on a stock server equipped with a single NVIDIA A6000 GPU. In terms of wall-clock time, fine-tuning took about 10 hours for each of the two models. Hence, comparatively, fine-tuning remains significantly more resource-intensive than the BVARs and factor models we considered, which can be trained on standard CPUs in a fraction of the time.[22]

Figure 9 shows the comparison of zero-shot and fine-tuned performance for TimesFM and Moirai-large. We observe that fine-tuning yields results that are similar or slightly better than the zero-shot performance for both models. This result overall isn't particularly surprising for our setup: a large model and a small dataset. While fine-tuning offers the potential for adapting models to a domain, it comes with its own set of challenges, such as hyperparameter optimization and increased computational resource (e.g. GPUs) requirements, thus needs careful consideration. Given the marginal performance gains achieved through fine-tuning in our context, the inherent "plug-and-play" simplicity of zero-shot forecasting presents itself as a potentially more practical approach.

[18] Kumar et al. (2022) discusses how in these situations the model may also lose the knowledge acquired during pretraining, leading to the so-called catastrophic forgetting.

[19] Kumar et al. (2022) refers to this as Linear Probing.

[20] In the case of Moirai, we also consider fine-tuning using the continued-pretraining strategy, as this was the recommended approach provided by the authors in the model's repository, but in this case the model ended up severely overfitting and producing poor out-of-sample forecasts.

[21] Within each fine-tuning round, the training FRED-MD data was split into an 80:20 ratio, with 20% reserved for validation. We trained these models for 200 epochs, utilizing early stopping with a patience of 5 to mitigate overfitting. To further counter overfitting and ensure stable training on our limited dataset, we reduced the learning rate, preventing large weight updates.

[22] We also considered the continued-pretraining option for the same models, and in that case the total computing time was approximately 90 hours.

7 Conclusions

In this paper, we have provided a comprehensive evaluation of the forecasting performance of Time Series Language Models (TSLMs) in predicting macroeconomic variables. While recent advances in machine learning have led to the development of these models as promising alternatives to traditional econometric methods, our results indicate that their performance remains mixed. Specifically, only two of the five models we evaluated, Salesforce's Moirai and Google's TimesFM, consistently outperform a simple autoregressive benchmark. Furthermore, these same models struggle to deliver consistently superior forecasts compared to established macroeconomic forecasting methods such as Bayesian Vector Autoregressions (BVARs) and Factor Models. Our findings therefore suggest that while TSLMs can offer valuable insights, they are not yet a clear replacement for state-of-the-art econometric models.

A key challenge in applying TSLMs to macroeconomic forecasting is the lack of control over their training data. For example, many of these models are pretrained on the FRED-MD dataset, hence already contain the macroeconomic series that are the focus of our forecasting exercise, introducing potential biases and making it difficult to conduct a clean pseudo out-of-sample forecasting exercise. Furthermore, practical challenges and computational constraints make retraining these models in real time a non-trivial task. Taken together, these issues raise important concerns about the role of TSLMs in economic forecasting, particularly in comparison to existing time series methods such as BVARs and factor models which have proven extremely flexible and successful in these same settings.

Despite these limitations, our results indicate that TSLMs exhibit some promising features, particularly in capturing nonlinearities and adapting to evolving economic conditions. Their relatively stronger performance in the post-Covid-19 period suggests that they may have advantages in handling structural breaks and rapidly changing economic environments. As research in this area continues to evolve, it may be worthwhile exploring hybrid approaches that integrate the flexibility and generality of foundational models with the theoretical rigor of econometric methods. For example, it would be interesting to investigate whether incorporating structured economic constraints into the TSLMs' architectures could enhance their reliability. We leave this topic to further research.

References

Achiam, J., S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. (2023). Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Ansari, A. F., L. Stella, C. Turkmen, X. Zhang, P. Mercado, H. Shen, O. Shchur, S. S. Rangapuram, S. P. Arango, S. Kapoor, et al. (2024). Chronos: Learning the language of time series. *arXiv preprint arXiv:2403.07815*.

Banbura, M., D. Giannone, and L. Reichlin (2010). Large Bayesian vector autoregressions. *Journal of Applied Econometrics 25*(1), 71–92.

Bauwens, L., M. Lubrano, and J. Richard (2000). *Bayesian Inference in Dynamic Econometric Models*. Oxford University Press.

Bybee, L. (2023). Surveying generative ai's economic expectations. *arXiv preprint arXiv:2305.02823*.

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

Carriero, A., T. E. Clark, and M. Marcellino (2016). Common drifting volatility in large Bayesian VARs. *Journal of Business & Economic Statistics 34*(3), 375–390.

Carriero, A., T. E. Clark, and M. Marcellino (2019). Large Bayesian vector autoregressions with stochastic volatility and non-conjugate priors. *Journal of Econometrics 212*(1), 137–154.

Carriero, A., T. E. Clark, M. Marcellino, and E. Mertens (2022). Addressing COVID-19 outliers in BVARs with stochastic volatility. *Review of Economics and Statistics forthcoming*.

Carriero, A., G. Kapetanios, and M. Marcellino (2012). Forecasting government bond yields with large Bayesian vector autoregressions. *Journal of Banking and Finance 36*(7), 2026 – 2047.

Carvalho, C. M., P. Lima, and H. F. Lopes (2024). Large Bayesian Additive Vector Autoregressive Tree Models. Available at https://hedibert.org/wp-content/uploads/2024/12/Hedibert-COBAL-EBEB-slides.pdf.

Chan, J. C. C. (2022). Asymmetric conjugate priors for large bayesian vars. *Quantitative Economics 13*(3), 1145–1169.

Chen, J., G. Tang, G. Zhou, and W. Zhu (2025). ChatGPT and Deepseek: Can They Predict the Stock Market and Macroeconomy? Available at https://arxiv.org/pdf/2502.10008.

Chen, Y., B. T. Kelly, and D. Xiu (2022). Expected returns and large language models. Available at SSRN: https://ssrn.com/abstract=4416687.

Chipman, H. A., E. I. George, and R. E. McCulloch (2010). BART: Bayesian additive regression trees. *The Annals of Applied Statistics 4*(1), 266 – 298.

Clark, T. E., F. Huber, G. Koop, M. Marcellino, and M. Pfarrhofer (2023). Tail forecasting with multivariate Bayesian additive regression trees. *International Economic Review 64*(3), 979–1022.

Clark, T. E. and M. W. McCracken (2012). Reality checks and comparisons of nested predictive models. *Journal of Business & Economic Statistics 30*(1), 53–66.

Clark, T. E. and M. W. McCracken (2015). Nested forecast model comparisons: A new approach to testing equal accuracy. *Journal of Econometrics 186*(1), 160–177.

Das, A., W. Kong, R. Sen, and Y. Zhou (2024). A decoder-only foundation model for time-series forecasting. In *International Conference on Machine Learning*. PMLR.

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

Doan, T., R. Litterman, and C. Sims (1984). Forecasting and conditional projection using realistic prior distributions. *Econometric Reviews 3*(1), 1–100.

Ekambaram, V., A. Jati, N. Nguyen, P. Sinthong, and J. Kalagnanam (2023). Tsmixer: Lightweight mlp-mixer model for multivariate time series forecasting. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 459–469.

Ekambaram, V., A. Jati, N. H. Nguyen, P. Dayama, C. Reddy, W. M. Gifford, and J. Kalagnanam (2024). Ttms: Fast multi-level tiny time mixers for improved zero-shot and few-shot forecasting of multivariate time series. *arXiv preprint arXiv:2401.03955*.

Faria-e Castro, M. and F. Leibovici (2024). Artificial intelligence and inflation forecasts. Federal Reserve Bank of St. Louis Working Paper Series.

Garza, A. and M. Mergenthaler-Canseco (2023). TimeGPT-1. *arXiv preprint arXiv:2310.0358*.

Giannone, D., M. Lenza, and G. E. Primiceri (2015). Prior selection for vector autoregressions. *Review of Economics and Statistics 97*(2), 436–451.

Godahewa, R., C. Bergmeir, G. I. Webb, R. J. Hyndman, and P. Montero-Manso (2021). Monash time series forecasting archive. *arXiv preprint arXiv:2105.06643*.

Goswami, M., K. Szafer, A. Choudhry, Y. Cai, S. Li, and A. Dubrawski (2024). Moment: A family of open time-series foundation models. In *International Conference on Machine Learning*. PMLR.

Han, Z., C. Gao, J. Liu, J. Zhang, and S. Q. Zhang (2024). Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv preprint arXiv:2403.14608*.

Harvey, D., S. Leybourne, and P. Newbold (1997). Testing the equality of prediction mean squared errors. *International Journal of Forecasting 13*(2), 281 – 291.

Hu, E. J., Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, et al. (2022). Lora: Low-rank adaptation of large language models. *ICLR 1*(2), 3.

Huber, F., G. Koop, L. Onorante, M. Pfarrhofer, and J. Schreiner (2023). Nowcasting in a pandemic using non-parametric mixed frequency vars. *Journal of Econometrics 232*(1), 52–69.

Huber, F. and L. Rossini (2022). Inference in Bayesian additive vector autoregressive tree models. *The Annals of Applied Statistics 16*(1), 104 – 123.

Iglesias, G., E. Talavera, Á. González-Prieto, A. Mozo, and S. Gómez-Canaval (2023). Data augmentation techniques in time series domain: a survey and taxonomy. *Neural Computing and Applications 35*(14), 10123–10145.

Jiang, J., C. Han, W. Jiang, W. X. Zhao, and J. Wang (2023). Libcity: A unified library towards efficient and comprehensive urban spatial-temporal prediction. *arXiv preprint arXiv:2304.14343*.

Jin, M., S. Wang, L. Ma, Z. Chu, J. Y. Zhang, X. Shi, P.-Y. Chen, Y. Liang, Y.-F. Li, S. Pan, and Q. Wen (2024). Time-LLM: Time series forecasting by reprogramming large language models. In *International Conference on Learning Representations (ICLR)*.

Kadiyala, K. R. and S. Karlsson (1993). Forecasting with generalized bayesian vector auto regressions. *Journal of Forecasting 12*(3-4), 365–378.

Kadiyala, K. R. and S. Karlsson (1997). Numerical methods for estimation and inference in bayesian var-models. *Journal of Applied Econometrics 12*(2), 99–132.

Kim, A. G., M. Muhn, and V. V. Nikolaev (2024). Financial statement analysis with large language models. Available at SSRN: https://ssrn.com/abstract=4835311.

Koop, G., D. Korobilis, and D. Pettenuzzo (2019). Bayesian compressed vector autoregressions. *Journal of Econometrics 210*(1), 135–154. Annals Issue in Honor of John Geweke "Complexity and Big Data in Economics and Finance: Recent Developments from a Bayesian Perspective".

Korobilis, D. and D. Pettenuzzo (2019). Adaptive hierarchical priors for high-dimensional vector autoregressions. *Journal of Econometrics 212*(1), 241–271. Big Data in Dynamic Predictive Econometric Modeling.

Kumar, A., A. Raghunathan, R. Jones, T. Ma, and P. Liang (2022). Fine-tuning can distort pretrained features and underperform out-of-distribution. *arXiv preprint arXiv:2202.10054*.

Lenza, M. and G. E. Primiceri (2022). How to estimate a vector autoregression after March 2020. *Journal of Applied Econometrics 37*(4), 688–699.

Lim, B. and S. Zohren (2021). Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A 379*(2194), 20200209.

Litterman, R. B. (1986). Forecasting with bayesian vector autoregressions: Five years of experience. *Journal of Business and Economic Statistics 4*(1), pp. 25–38.

Lu, K., A. Grover, P. Abbeel, and I. Mordatch (2022, Jun.). Frozen pretrained transformers as universal computation engines. *Proceedings of the AAAI Conference on Artificial Intelligence 36*(7), 7628–7636.

McCracken, M. W. and S. Ng (2016). Fred-md: A monthly database for macroeconomic research. *Journal of Business & Economic Statistics 34*(4), 574–589.

Prüser, J. (2019). Forecasting with many predictors using bayesian additive regression trees. *Journal of Forecasting 38*(7), 621–631.

Rasul, K., A. Ashok, A. R. Williams, H. Ghonia, R. Bhagwatkar, A. Khorasani, M. J. D. Bayazi, G. Adamopoulos, R. Riachi, N. Hassen, M. Biloš, S. Garg, A. Schneider, N. Chapados, A. Drouin,

V. Zantedeschi, Y. Nevmyvaka, and I. Rish (2024). Lag-llama: Towards foundation models for probabilistic time series forecasting.

Schorfheide, F. and D. Song (2021). Real-time forecasting with a (standard) mixed-frequency VAR during a pandemic. NBER Working Paper 29535, National Bureau of Economic Research, Inc.

Sims, C. A. (1993). A nine-variable probabilistic macroeconomic forecasting model. In J. H. Stock and M. W. Watson (Eds.), *Business Cycles, Indicators, and Forecasting*, NBER Chapters, pp. 179–212. National Bureau of Economic Research, Inc.

Sims, C. A. and T. Zha (1998). Bayesian methods for dynamic multivariate models. *International Economic Review 39*(4), 949–968.

Stock, J. H. and M. W. Watson (2002, April). Macroeconomic Forecasting Using Diffusion Indexes. *Journal of Business & Economic Statistics 20*(2), 147–162.

Stock, J. H. and M. W. Watson (2006). Forecasting with many predictors. Volume 1 of *Handbook of Economic Forecasting*, pp. 515 – 554. Elsevier.

Torres, J. F., D. Hadjout, A. Sebaa, F. Martínez-Álvarez, and A. Troncoso (2021). Deep learning for time series forecasting: a survey. *Big Data 9*(1), 3–21.

Touvron, H., L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al. (2023). Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin (2017). Attention is all you need. *Advances in neural information processing systems 30*.

Wen, Q., T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, and L. Sun (2022). Transformers in time series: A survey. *arXiv preprint arXiv:2202.07125*.

Woo, G., C. Liu, A. Kumar, C. Xiong, S. Savarese, and D. Sahoo (2024). Unified training of universal time series forecasting transformers. *arXiv preprint arXiv:2402.02592*.

Xue, H. and F. D. Salim (2023). Promptcast: A new prompt-based learning paradigm for time series forecasting. *IEEE Transactions on Knowledge and Data Engineering*, 1–14.

Zellner, A. (1971). *An introduction to Bayesian inference in econometrics*. John Wiley & Sons, Ltd.

Appendix A Details on Priors

A.1 Prior Moments of Natural Conjugate Prior

We elicit Φ_0 and Ω_0 in such a way that the resulting moments of the matrices $\Phi_l, l = c, 1, \ldots, p$ are as follows:

$$E[\Phi_l^{(ij)}] = \begin{cases} \Phi^* & \text{if } i = j, l = 1 \\ 0 & \text{otherwise} \end{cases}, \; Var[\Phi_l^{(ij)}] = \begin{cases} \frac{\lambda_1^2}{l^2}\frac{\sigma_i^2}{\hat{\sigma}_j^2}, \; l = 1, ..., p, \\ \lambda_0^2 \sigma_i^2, \; l = c \end{cases} \tag{A.1}$$

where $\Phi_l^{(ij)}$ denotes the element in position (i, j) in the matrix Φ_l. The prior mean Φ^* is set to either 1 or 0 depending on the presumed order of integration of the series. For the intercept ($l = c$) we assume an uninformative prior with mean 0 and standard deviation $\lambda_0 \sigma_i$. The shrinkage parameter λ_1 measures the overall tightness of the prior: when $\lambda_1 \to 0$ the prior is imposed exactly and the data do not influence the estimates, while as $\lambda_1 \to \infty$ the prior becomes loose and the prior information does not influence the estimates, which will approach the standard OLS estimates. To set each scale parameter $\hat{\sigma}_j$ in (A.1) we follow common practice (see e.g. Litterman (1986); Sims and Zha (1998)) and set it equal to the standard error of regression from a univariate autoregressive model. The parameters σ_i^2 appearing in (A.1) are instead coming directly from the error variance Σ on which this prior is conditioning upon, trough the Kronecker multiplication shown in (4).[23]

The prior specification is completed by choosing v_0 and S_0. Following Kadiyala and Karlsson (1997) we set these in a way that they are as uninformative as possible and such that the prior expectation of the error variance Σ is finite and diagonal with diagonal elements given by the standard errors of regression of univariate autoregressive models, i.e. $E[\Sigma] = diag(\hat{\sigma}_1^2, ..., \hat{\sigma}_N^2)$. To achieve this we set $v_0 = N + 2$ and S_0 diagonal with diagonal elements $(v_0 - N - 1)\hat{\sigma}_j^2, j = 1, ...N$.

A.1.1 Dummy Initial Observations

Doan et al. (1984) and Sims (1993) have proposed to complement the priors described above with additional priors which favour unit roots and cointegration. Both these priors were motivated by the need to avoid having an unreasonably large share of the sample period variation in the data accounted for by deterministic components. They are both calibrated using the average of the first p observations in the sample and are typically implemented as dummy observations.

[23] Incidentally, note that such multiplication also induces correlation across coefficients belonging to different equations, which is not apparent in (A.1) because it focuses only on the variances of the individual coefficients.

The "sum of coefficients" prior expresses a belief that when the average of lagged values of a variable is at some level $\bar{y}_{0i}$, that same value $\bar{y}_{0i}$ is likely to be a good forecast of future observations, and is implemented by augmenting the system in (3) with the dummy observations Y_{d_1} and X_{d_1} with generic elements:

$$y_d(i,j) = \begin{cases} \bar{y}_{0i}/\mu_1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \; ; \; x_d(i,s) = \begin{cases} \bar{y}_{0i}/\mu_1 & \text{if } i = j, s < M \\ 0 & \text{otherwise,} \end{cases} \tag{A.2}$$

where i and j go from 1 to N while s goes from 1 to K. When $\mu_1 \to 0$ the model tends to a form that can be expressed entirely in terms of differenced data, there are as many unit roots as variables and there is no cointegration.

The "single unit root" prior introduces a single dummy observation such that all values of all variables are set equal to the corresponding averages of initial conditions up to a scaling factor $(1/\lambda_4)$. It is implemented by adding to the system in (3) the dummy variables Y_{d_2} and X_{d_2} with generic elements:

$$y_d(j) = \bar{y}_{0j}/\mu_2; \;\; x_d(s) = \begin{cases} \bar{y}_{0j}/\mu_2 & \text{for } s < M \\ 1/\mu_2 & \text{for } \; s = M, \end{cases} \tag{A.3}$$

where j goes from 1 to N while s goes from 1 to K. As $\mu_2 \to 0$ the model tends to a form in which either all variables are stationary with means equal to the sample averages of the initial conditions, or there are unit root components without drift terms, which is consistent with cointegration.

A.1.2 Hyperparameters

To make the prior operational, one needs to choose the value of the hyperparameters which control the overall tightness of the Minnesota, sum of coefficients, and single unit root priors. As mentioned above, in the natural conjugate NIW framework the joint priors and joint posteriors are all matricvariate t distributions, which implies that the marginal likelihood $p(Y)$ can be computed in closed form simply by inverting Bayes formula and consists in the ratio of the integrating constants of prior, likelihood, and posterior:

$$p(Y) = \pi^{\frac{-TN}{2}} \times \frac{|\Omega_1|^{\frac{N}{2}}}{|\Omega_0|^{\frac{N}{2}}} \times \frac{|\bar{S}|^{-\frac{v_0+T}{2}}}{|S_0|^{-\frac{v_0}{2}}} \times \frac{\Gamma_N(\frac{v_0+T}{2})}{\Gamma_N(\frac{v_0}{2})}, \tag{A.4}$$

where $\Gamma_N(\cdot)$ denotes the N-variate gamma function. A derivation based on theorem A.19 in Bauwens et al. (2000) can be found in Carriero et al. (2012).

We follow Carriero et al. (2012) and Korobilis and Pettenuzzo (2019) and choose the vector of hyperparameters θ by maximizing the marginal data density: $\theta^* = \arg\max_\theta \ln p(Y)$.[24] As detailed above, the vector θ comprises the parameters λ_1, μ_1, and μ_2 that control the overall tightness of the Minnesota, sum of coefficients, and single unit root priors, respectively.

A.2 Prior Moments for Asymmetric Conjugate Prior

We set we $m_{\beta,i} = 0$ to shrink the VAR coefficients to zero for data expressed either in first-differenced or growth rates, while for data in levels or log-levels, $m_{\beta,i}$ is set to be zero as well except the coefficient associated with the first own lag, which is set to be one. As for $V_{\beta,i}$, we take the following approach

$$V_{\beta,i} = \begin{cases} \frac{\kappa_1}{l^2 s_i^2} & \text{for the coefficient on the } l\text{-th lag of variable } i \\ \frac{\kappa_2}{l^2 s_j^2} & \text{for the coefficient on the } l\text{-th lag of variable } j \ \ (j \neq i) \\ \kappa_3 & \text{for the intercept.} \end{cases} \tag{A.5}$$

The hyperparameter κ_1 controls the overall shrinkage strength for coefficients on own lags, whereas κ_2 controls those on lags of other variables. Both will be selected by maximizing the associated marginal likelihood. We set $\kappa_3 = 100$, which implies essentially no shrinkage for the intercepts.[25] Note this prior variance allows for cross-shrinkage, i.e. the parameter κ_2 can be different than 1, which implies the degree of shrinkage can be different for lags of variables that are not the dependent variable of a given equation. This is therefore more general than the prior variance implied by the natural conjugate prior appearing in (A.1). The latter can be obtained as a special case by setting $\kappa_2 = 1$ (i.e. ruling out cross-shrinkage).

We set the prior degrees of freedom v_0 and scale s_i^2 as in the natural conjugate case, i.e. in a way that they are as uninformative as possible and such that the prior expectation of the error variance Σ is finite and diagonal with diagonal elements given by the standard errors of regression of univariate autoregressive models, i.e. $E[\Sigma] = diag(\hat{\sigma}_1^2, ..., \hat{\sigma}_N^2)$.

[24] Giannone et al. (2015) have developed a hierarchical Bayes approach in which a prior is elicited for θ and its posterior distribution is derived via a random walk Metropolis step. This approach is more sophisticated but it requires MCMC methods, a feature that is less desirable when handling very large datasets. In our experience with the FRED-MD dataset, this approach and the simple maximization of the marginal likelihood produce very similar results.

[25] Note that in this approach we are setting our priors on the VAR coefficients from the structural representation of the model. It would also be possible to set prior means and variances directly on the reduced-form coefficients, and then back-out the implied prior moments of the structural-form coefficients.

Appendix B Data and Transformations

Abbreviation	Description	Transf. code	Medium	Large	X-large
PAYEMS	All Employees: Total nonfarm	5	x	x	x
INDPRO	IP Index	5	x	x	x
FEDFUNDS	Effective Federal Funds Rate	1	x	x	x
UNRATE	Civilian Unemployment Rate	1	x	x	x
RPI	Real personal income	5	x	x	x
DPCERA3M086SBEA	Real PCE	5	x	x	x
CMRMTSPLx	Real Manu. and TradeIndustries Sales	5	x	x	x
CUMFNS	Capacity Utilization: Manufacturing	1	x	x	x
CES0600000007	Avg Weekly Hours: Goods-Producing	1	x	x	x
HOUST	Housing Starts, Total	4	x	x	x
S&P 500	S&P's Common Stock Price Index: Composite	5	x	x	x
T1YFFM	1-Year Treasury C Minus FEDFUNDS	1	x	x	x
T10YFFM	10-Year Treasury C Minus FEDFUNDS	1	x	x	x
BAAFFM	Moodys Baa Corporate Bond Minus FEDFUNDS	1	x	x	x
EXUSUKx	U.S.-UK Foreign Exchange Rate	5	x	x	x
WPSFD49207	PPI: Final Demand: Finished Goods	5	x	x	x
PPICMM	PPI: Metals and metal products	5	x	x	x
PCEPI	Personal Consumption Expenditures	5	x	x	x
CES0600000008	Avg Hourly Earnings: Goods-Producing	6	x	x	x
HWI	Help-Wanted Index for U	1		x	x
HWIURATIO	Help Wanted to Unemployed ratio	1		x	x
CLF16OV	Civilian Labor Force	5		x	x
M1SL	M1 Money Stock	5		x	x
M2SL	M2 Money Stock	5		x	x
M2REAL	Real M2 Money Stock	5		x	x
S&P div yield	S&P's Composite Common Stock: Dividend Yield	1		x	x
S&P PE ratio	S&P's Common Stock: Price/Earnings ratio	5		x	x
TB6MS	6-Month Treasury Bill	1		x	x
GS1	1-Year Treasury Rate	1		x	x
GS5	5-Year Treasury Rate	1		x	x
AAA	Moody's Seasoned Aaa Corporate Bond Yield	1		x	x
BAA	Moody's Seasoned Baa Corporate Bond Yield	1		x	x
OILPRICEx	Crude Oil, spliced WTI and Cushing	5		x	x
INVEST	Securities in Bank Credit at All Commercial Banks	5		x	x
W875RX1	Real personal income ex transfer receipts	5			x
RETAILx	Retail and Food Services Sales	5			x
IPFPNSS	IP: Final Products	5			x
IPFINAL	IP: Final Products (Market Group)	5			x
IPCONGD	IP: Consumer Goods	5			x
IPDCONGD	IP: Durable Consumer Goods	5			x
IPNCONGD	IP: Non-Durable Consumer Goods	5			x
IPBUSEC	IP: Business & Equipment	5			x

IPMAT	IP: Materials	5	x
IPDMAT	IP: Durable Goods Materials	5	x
IPNMAT	IP: Non-Durable Goods Materials	5	x
IPMANSICS	IP: Manufacturing (SIC)	5	x
IPB51222S	IP: Residential Utilities	5	x
IPFUELS	IP: Fuels	5	x
CE16OV	Civilian Employment	5	x
UEMPMEAN	Average Duration of Unemployment (Weeks)	1	x
UEMPLT5	Civilians Unemployed: Less Than 5 Weeks	5	x
UEMP5TO14	Civilians Unemployed for 5-14 Weeks	5	x
UEMP15OV	Civilians Unemployed: 15 Weeks & Over	5	x
UEMP15T26	Civilians Unemployed for 15-26 Weeks	5	x
UEMP27OV	Civilians Unemployed for 27 Weeks and Over	5	x
CLAIMSx	Initial Claims	5	x
USGOOD	All Employees: Goods-Producing Industries	5	x
CES1021000001	All Employees: Mining and Logging: Mining	5	x
USCONS	All Employees: Construction	5	x
MANEMP	All Employees: Manufacturing	5	x
DMANEMP	All Employees: Durable goods	5	x
NDMANEMP	All Employees: Nondurable goods	5	x
SRVPRD	All Employees: Service-Providing Industries	5	x
USTPU	All Employees: TT&U	5	x
USWTRADE	All Employees: Wholesale Trade	5	x
USTRADE	All Employees: Retail Trade	5	x
USFIRE	All Employees: Financial Activities	5	x
USGOVT	All Employees: Government	5	x
AWOTMAN	Avg Weekly Overtime Hourse: Manufacturing	1	x
AWHMAN	Avg Weekly Hours: Manufacturing	1	x
HOUSTNE	Housing Starts, Northeast	4	x
HOUSTMW	Housing Starts, Midwest	4	x
HOUSTS	Housing Starts, South	4	x
HOUSTW	Housing Starts, West	4	x
PERMIT	New Private Housing Permits (SAAR)	4	x
PERMITNE	New Private Housing Permits, Northeast (SAAR)	4	x
PERMITMW	New Private Housing Permits, Midwest (SAAR)	4	x
PERMITS	New Private Housing Permits, South (SAAR)	4	x
PERMITW	New Private Housing Permits, West (SAAR)	4	x
AMDMNOx	New Orders for Durable goods	5	x
AMDMUOx	Unfilled Orders for Durable goods	5	x
BUSINVx	Total Business Inventories	5	x
ISRATIOx	Total Business: Inventories to Sales Ratio	1	x
BOGMBASE	Monetary Base: Total	5	x
TOTRESNS	Total Reserves of Depository Institutions	5	x
BUSLOANS	Commercial and Industrial Loans, All Commercial Banks	5	x
REALLN	Real Estate Loans	5	x
NONREVSL	Total Nonrevolving Credit	5	x

CONSPI	Credit to PI ratio	1		x
CP3Mx	3-Month AA Financial Commercial Paper Rate	1		x
TB3MS	3-Month Treasury Bill	1		x
GS10	10-Year Treasury Rate	1		x
COMPAPFFx	3-Month Commercial Paper Minus FEDFUNDS	1		x
TB3SMFFM	3-Month Treasury C Minus FEDFUNDS	1		x
TB6SMFFM	6-Month Treasury C Minus FEDFUNDS	1		x
T5YFFM	5-Year Treasury C Minus FEDFUNDS	1		x
AAAFFM	Moodys Aaa Corporate Bond Minus FEDFUNDS	1		x
EXSZUSx	Switzerland-U.S. Foreign Exchange Rate	5		x
EXJPUSx	Japan-U.S. Foreign Exchange Rate	5		x
EXCAUSx	Canada-U.S. Foreign Exchange Rate	5		x
WPSFD49502	PPI: Final Demand	5		x
WPSID61	PPI: Processed goods	5		x
WPSID62	PPI: Unprocessed goods	5		x
CPIAUCSL	CPI: All Items	5		x
CPIAPPSL	CPI: Apparel	5		x
CPITRNSL	CPI: Transportation	5		x
CPIMEDSL	CPI: Medical Care	5		x
CUSR0000SAC	CPI: Commodities	5		x
CUSR0000SAD	CPI: Durables	5		x
CUSR0000SAS	CPI: Services	5		x
CPIULFSL	CPI: All Items Less Food	5		x
CUSR0000SA0L2	CPI: All Items Less Shelter	5		x
CUSR0000SA0L5	CPI: All Items Less Medical Care	5		x
DDURRG3M086SBEA	PCE: Durable goods	5		x
DNDGRG3M086SBEA	PCE: Nondurable goods	5		x
DSERRG3M086SBEA	PCE: Services	5		x
CES2000000008	Avg Hourly Earnings: Construction	5		x
CES3000000008	Avg Hourly Earnings: Manufacturing	5		x
DTCOLNVHFNM	Consumer Motor Vehicle Loans Outstanding	5		x
DTCTHFNM	Total Consumer Loans and Leases	5		x

Notes: The dataset described in McCracken and Ng (2016) is available for download at https://research.stlouisfed.org/econ/. Column Transf. code indicates the transformation of a series x_t, where: (1) no transformation, (2) Δx_t, (5) $\Delta \log(x_t)$, (6) $\Delta^2 \log(x_t)$ with Δ^i indicating i th differences. Columns Medium, Large, and X-large refer to the different model sizes discussed in Section 3.