International Journal of Forecasting 40 (2024) 942–957

Contents lists available at ScienceDirect

International Journal of Forecasting

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

A multi-task encoder-dual-decoder framework for mixed frequency data prediction

Jiahe Lin[a], George Michailidis[b,*]

[a] Machine Learning Research, Morgan Stanley, New York, 10036, NY, USA
[b] Department of Statistics, University of California, Los Angeles, Los Angeles, 90095, CA, USA

ARTICLE INFO

Keywords:
Mixed-frequency data
Encoder–decoder
LSTM
Transformer
Multi-horizon forecasts
Nowcasting

ABSTRACT

Mixed-frequency data prediction tasks are pertinent in various application domains, in which one leverages progressively available high-frequency data to forecast/nowcast the low-frequency ones. Existing methods in the literature tailored to such tasks are mostly linear in nature; depending on the specific formulation, they largely rely on the assumption that the (latent) processes that govern the dynamics of the high- and low-frequency blocks of variables evolve at the same frequency, either the low or the high one. This paper develops a neural network-based multi-task shared-encoder-dual-decoder framework for joint multi-horizon prediction of both the low- and high-frequency blocks of variables, wherein the encoder/decoder modules can be either long short-term memory or transformer ones. It addresses forecast/nowcast tasks in a unified manner, leveraging the encoder–decoder structure that can naturally accommodate the mixed-frequency nature of the data. The proposed framework exhibited competitive performance when assessed on both synthetic data experiments and two real datasets of US macroeconomic indicators and electricity data.

1. Introduction

Mixed-frequency (MF) data arise in forecasting tasks across different domains. One popular application pertains to the prediction of key quarterly macroeconomic indicators (e.g., gross domestic product) by leveraging monthly economic and financial variables (e.g., Bell, Co, Stone, & Wallis, 2014; Cimadomo, Giannone, Lenza, Monti, & Sokol, 2022; Cross, Hou, & Poon, 2020; Huber, Koop, Onorante, Pfarrhofer, & Schreiner, 2023; Kuzin, Marcellino, & Schumacher, 2011). Other applications include tasks in environmental sciences, such as forecasting indicators that reflect the health of a lake environment based on high-frequency weather episodic events (Jennings et al., 2012), or hourly wind speed based on higher-frequency variables that capture weather conditions (Yang, Tian, & Hao, 2022). To handle the aforementioned forecasting tasks involving data observed at different frequencies, most existing methods in the literature are linear in nature and largely fall into the following three categories: regression-type models with a univariate response variable, vector autoregressive models, and dynamic factor models. The latter two have a multivariate response and the variables are modeled as a joint system. These methods are briefly reviewed in Section 1.1.

In recent years, neural network (NN)-based models have been increasingly used in forecasting applications. For example, multilayer perceptrons (MLPs), recurrent neural networks (RNNs), and its long short-term memory (LSTM) (Hochreiter & Schmidhuber, 1997) variant were used in the M3 and M4 forecasting competitions (Makridakis & Hibon, 2000;

* Corresponding author.
E-mail address: gmichail@ucla.edu (G. Michailidis).

https://doi.org/10.1016/j.ijforecast.2023.08.003

Makridakis, Spiliotis, & Assimakopoulos, 2020) and exhibited good performance vis-à-vis traditional statistical models (Makridakis, Wheelwright, & Hyndman, 2008), as a result of their expressiveness and the ability to handle complex nonlinear dynamics.

In this work, we propose to leverage NN architectures consisting of encoder and decoder modules (e.g., sequence-to-sequence (seq2seq) Sutskever, Vinyals, & Le, 2014 or transformers Vaswani et al., 2017) to model MF data through multi-task learning. Such architectures were originally developed for natural language processing tasks and have achieved state-of-the-art performance. Note that NN-based models have recently been employed to deal with two distantly related yet distinct tasks that involve asynchronous (Bao, Yue, & Rao, 2017; Binkowski, Marti, & Donnat, 2018; Lepot, Aubin, & Clemens, 2017; Li & Marlin, 2020) and multi-scale (Che, Purushotham, Cho, Sontag, & Liu, 2018; Shabani, Abdi, Meng, & Sylvain, 2022) time series data. A key difference between MF and asynchronous data is that the former are synchronous at low sampling frequencies, while in the latter case, each time series is sampled at irregular time points. Hence, the focus of statistical and machine learning methods for dealing with asynchronous data has primarily been on imputing missing values of the time series, in order to create a complete dataset of regularly spaced observations (see Weerakody, Wong, Wang, & Ela, 2021 for a recent review). In contrast to MF data, in multi-scale modeling, all time series are measured on the same frequency, and the main objective is to capture complex temporal dynamics that arise at different time scales (e.g., daily, weekly, and monthly) so as to improve forecasting performance. In other words, multi-scale time series modeling pertains to the assumption of the temporal dependencies of the time series, as opposed to having multiple time series sampled at different resolutions.

Next, we introduce necessary notation and concepts. There are d_y low-frequency (e.g., quarterly) variables and d_x high-frequency (e.g., monthly) ones for which data are collected over time. Let $\{\mathbf{y}_t \in \mathbb{R}^{d_y}\}$ denote the d_y-dimensional low-frequency block and $\{\mathbf{x}_t \in \mathbb{R}^{d_x}\}$ the d_x-dimensional high-frequency one; the subscript corresponds to a generic time index, unless otherwise specified. The frequency ratio r provides the relative granularity of these two collections; for example, in the case of quarterly/monthly time series, $r = 3$. Note that in the case where N_y observations of the low-frequency block are available, the number of available observations for the high-frequency block, denoted by N_x, satisfies $rN_y \leq N_x \leq (r+1)N_y$.[1] Further, let $\widehat{\mathbf{y}}_{N_y+1}$ denote the prediction at time period $(N_y + 1)$ for the low-frequency block, which can come from either a forecast or a nowcast, based on available information up to N_y for the low-frequency and N_x for the high-frequency blocks, that is, $\{\mathbf{y}_t\}_{t=1}^{N_y}$, $\{\mathbf{x}_t\}_{t=1}^{N_x}$. A forecast pertains to the case where $N_x \equiv rN_y$. In the case of a nowcast for the same time period, $N_x = rN_y + h$ for some $1 \leq h \leq r$. That is, for the quarterly measured gross domestic product in the period of March, a forecast is based on data available up to the preceding December, whereas a nowcast is based on data up to the preceding December for the quarterly variables, and up to the preceding January, February, or March for the monthly variables, depending on the corresponding vintage of the nowcast.

1 The case of $N_x = (r + 1)N_y$ could occur due to data publishing delays (usually for low-frequency variables) in certain applications areas. For example, the US gross domestic product is published with one month delay after the end of the corresponding quarter.

1.1. Related work on modeling mixed-frequency data

We first provide an overview of existing methods in the literature that address forecasting and nowcasting tasks involving MF data. These methods are mostly linear in nature, and can be broken into three categories based on how the low- and high-frequency blocks are handled, namely, whether the underlying dynamics of the system are assumed to be at the low or high frequency.

In the first category, the mixed data sampling (MIDAS) regression framework (Ghysels, Sinko, & Valkanov, 2007) models a low-frequency univariate response as a function of its own temporal lags and high-frequency predictors' contemporaneous and lagged values. The specification is essentially agnostic to the frequency of the data. Note that the presence of many lags in the high-frequency variables leads to a proliferation of regression coefficients, and hence various combinations of restrictions and weighting schemes have been developed to reduce the effective number of model parameters to be estimated from the data. Neural network-based variants of the model have also been investigated in recent years (Xu, Liu, Jiang, & Zhuo, 2021; Xu, Zhuo, Jiang, & Liu, 2019).

The second category postulates that the system evolves based on a vector autoregressive (VAR) model at the low frequency, and thus requires frequency alignment of the high-frequency block (Ghysels, 2016; McCracken, Owyang, & Sekhposyan, 2015). Specifically, each individual high-frequency time series is "expanded" into r new ones, with r being the frequency ratio. The ith ($i = 1, \ldots, r$) new time series is obtained by thinning the original process with every rth observation extracted starting from index i. This expansion leads to a proliferation of model parameters, and penalized approaches have been developed for estimation and inference purposes (Chakraborty, Khare, & Michailidis, 2023; Uematsu & Tanaka, 2019).

The third category assumes that the underlying time series that drive the dynamics of the system evolve at the high frequency, and how the low-frequency block is modeled depends on the specific approach adopted. In the case of VAR models, the high- and low-frequency blocks evolve as a joint system, and the unobserved values for the low-frequency variables are treated as missing and are imputed; see Ankargren, Unosson, and Yang (2020), Foroni and Marcellino (2014), Gefang, Koop, and Poon (2020), Mariano and Murasawa (2003) and Schorfheide and Song (2015). Another stream of work assumes that the dynamics of the system are governed by a few factors, with the latter possessing autoregressive dynamics. Factors are extracted from the high-frequency block, with their dynamics estimated based on a state-space representation. Subsequently, the factors are used as predictors in a regression model with the low-frequency variables being the responses, where the factors are either aggregated at the low frequency (Foroni & Marcellino, 2013), or their values that correspond to the low-frequency timestamps are used (Giannone, Reichlin, & Small, 2008).

Next, we briefly review NN-based methods that handle asynchronous time series given their conceptual similarity to MF data. The predominant strategy is to first convert the irregularly spaced dataset into a "regular" one by imputing the values that are missing at a regularly spaced grid. Various NN-based models have been used for the imputation task, including unidirectional or bidirectional RNNs/LSTMs (Kim & Chi, 2018; Ma & Leung, 2019) and their variants, whether enhanced with an attention mechanism (Dabrowski & Rahman, 2019; Shukla & Marlin, 2021) or coupled with ordinary differential equations (Schirmer, Eltayeb, Lessmann, & Rudolph, 2021). Unsupervised approaches based on generative models such as variational autoencoders and generative adversarial networks (Li & Marlin, 2020; Luo, Cai, Zhang, Xu, et al., 2018) are also considered. In Baytas et al. (2017) amd Lipton, Kale, and Wetzel (2016), the authors construct additional features based on whether the values are missing (0/1) and the time elapsed between available observations. In Che et al. (2018), the time intervals between available observations are used as weights to determine the relevancy of the observed values to the missing ones.

Summary of contributions. We introduce a multi-task shared-encoder-dual-decoder framework to handle MF data, where the dynamics of the low- and high-frequency blocks of time series are learned simultaneously, based on which their predictions can be obtained. Depending on the encoder/decoder modules used, the framework can be either an LSTM or a transformer-based one: the first considers an LSTM-based encoder–decoder architecture with temporal local attention, wherein both the encoding and the decoding occur sequentially; the second considers a transformer-based architecture, where past information is ingested and processed in a parallel fashion, leveraging a self-attention mechanism. The proposed framework has a flexible modeling structure that does not impose specific assumptions on the resolution of the underlying process or processes that govern the high- and low-frequency blocks. Such flexibility is further reflected in the following two aspects. First, the framework handles forecasting and nowcasting tasks in a unified manner, without requiring frequency alignment or model re-specification to accommodate the prediction vintages of interest. Appendix A elaborates on how this is needed for certain existing approaches. Second, the framework does not posit specific assumptions on how the low-frequency block evolves. Consequently, it does not require an imputation or aggregation mechanism for the block, in contrast to selected existing approaches.

The remainder of the paper is organized as follows. In Section 2, we provide a brief review of the building blocks used in the proposed framework. Section 3 describes the proposed framework in detail, including model architectures, training, and prediction. Model performance vis-à-vis competing approaches is assessed on synthetic datasets in Section 4 and on a real dataset of key US macroeconomic indicators in Section 5. An additional application to electricity data is briefly presented in Appendix C. Finally, Section 6 concludes the paper.

2. Preliminaries

In this section, we provide a brief overview of the building blocks of the proposed modeling framework. Throughout this section, we use $\mathbf{x}$ to denote the encoder inputs and $\mathbf{y}$ for the decoder inputs/outputs; and T_x and T_y respectively correspond to the encoder and decoder context (i.e., look-back history) length. The problem under consideration is to model the following conditional probability:

$$\mathbb{P}\big(\mathbf{y}_1, \dots, \mathbf{y}_{T_y} | \mathbf{x}_1, \dots, \mathbf{x}_{T_x}\big).$$

Note that in the natural language processing domain where these encoder/decoder modules were originally developed, the subscript indices in the corresponding input/output sequences do not correspond to physical timestamps; rather, they usually correspond to the order that words appear in sentences. For example, in neural machine translation, $\{\mathbf{x}_t\}_{t=1}^{T_x}$ corresponds to some input sentence that needs to be translated into a sentence in another language, which is denoted by $\{\mathbf{y}_t\}_{t=1}^{T_y}$. We keep this convention in this section and use t in subscripts as the generic indexing of the steps. In Section 3, we further elaborate how these building blocks are incorporated in the proposed MF data forecasting framework and make the dependency on physical timestamps of the observed data explicit.

2.1. Temporal attention-based LSTM encoder–decoder

LSTM encoder–decoder. The architecture of an RNN encoder–decoder was introduced in a seminal paper (Cho et al., 2014) for phase-based statistical machine translation and is widely used in seq2seq learning tasks (Sutskever et al., 2014). The architecture consists of two RNNs: the encoder one that reads in and encodes the variable-length input sequence $\{\mathbf{x}_t\}_{t=1}^{T_x}$ into a fixed-length vector $\mathbf{s}_0$ that contains the summary "state" information of the whole input; and the decoder one that is trained to sequentially generate the output $\mathbf{y}_t, t = 1, \dots, T_y$, based on both $\mathbf{s}_0$ and $\mathbf{y}_{t-1}$.

Specifically, consider a multivariate input sequence $\mathbf{x}_t, t = 1, \dots, T_x$. At the encoding stage, through a non-linear function f^h, $\mathbf{x}_t$ is sequentially encoded into $\mathbf{h}_t$, which denotes the hidden states of the encoder RNN. That is,

$$\mathbf{h}_t = f^h(\mathbf{x}_t, \mathbf{h}_{t-1}).$$

The terminal state $\mathbf{h}_{T_x} =: \mathbf{s}_0$ is considered the carrier of the input information to be used in decoding. At the decoding stage, for $t = 1, \dots, T_y$ the conditional probability of the output sequence is modeled through a non-linear function $g(\cdot)$. That is,

$$\mathbb{P}\big(\mathbf{y}_t | \{\mathbf{y}_1, \dots, \mathbf{y}_{t-1}\}, \{\mathbf{x}_i\}_{i=1}^{T_x}\big) = g\big(\mathbf{s}_t\big),$$

$$\mathbf{s}_t = f^s(\mathbf{s}_{t-1}, \mathbf{y}_{t-1}; \mathbf{s}_0),$$

where $\mathbf{s}_t$ ($t > 0$) corresponds to decoder hidden states. Note that we keep $\mathbf{s}_0$ to highlight the decoder's implicit dependency on the encoder terminal state, although for decoding step t ($t > 1$), such dependency is no longer direct. Both f^h and f^s usually correspond to LSTM (Hochreiter & Schmidhuber, 1997) or GRU (Cho et al., 2014) cells; in what follows, we use RNN and LSTM interchangeably to refer to recurrent networks of such sequential nature. With $\mathbf{s}_0$ as the connector, the two RNNs are trained jointly to minimize some cost function.

Temporal attention. At the decoding stage, instead of having decoder hidden states relying on a single state vector, a temporal attention mechanism (Bahdanau, Cho, & Bengio, 2014) enables them to access all hidden states $\{\mathbf{h}_t\}_{t=1}^{T_x}$ from the encoder through some context vector $\mathbf{c}_t$. That is,

$$\mathbf{s}_t = f^s(\mathbf{c}_t, \mathbf{s}_{t-1}, \mathbf{y}_{t-1}),$$

$$\mathbf{c}_t = \sum_{t'=1}^{T_x} \alpha_{tt'}\mathbf{h}_{t'}, \quad \text{where} \quad \alpha_{tt'} := \exp(e_{tt'}) / \sum_{t'=1}^{T_x} \exp(e_{tt'}).$$

The context vectors ($\mathbf{c}_t$) are linear combinations of all encoder hidden states, with the weights obtained by applying a softmax operator on the "energies" $e_{tt'}$. The latter $e_{tt'} := a(\mathbf{s}_{t-1}, \mathbf{h}_{t'})$ are learned through some alignment model $a(\cdot)$, which is typically a small and shallow feed forward network. The alignment model is jointly trained with the encoder and decoder RNNs.

The above exposition provides details on the inner structure and workings of the encoder and decoder, and can be further abstracted to the level of two mappings that provide useful representations for future technical developments. Specifically, the encoder maps its input sequence $\{\mathbf{x}_t\}_{t=1}^{T_x}$ into their corresponding hidden states $\{\mathbf{h}_t\}_{t=1}^{T_x}$. That is,

$$\Phi : \{\mathbf{x}_1, \ldots, \mathbf{x}_{T_x}\} \mapsto \{\mathbf{h}_1, \ldots, \mathbf{h}_{T_x}\}. \tag{1}$$

The decoder takes $\{\mathbf{y}_t\}_{t=0}^{T_y-1}$ (decoder input sequence) and $\{\mathbf{h}_t\}_{t=1}^{T_x}$ as inputs and maps them into state vectors $\{\mathbf{s}_t\}$. That is,

$$\Psi : \{\mathbf{y}_0, \ldots, \mathbf{y}_{T_y-1}, \{\mathbf{h}_t\}_{t=1}^{T_x}\} \mapsto \{\mathbf{s}_1, \ldots, \mathbf{s}_{T_y}\}, \tag{2}$$

with $\{\mathbf{y}_0, \ldots, \mathbf{y}_{T_y-1}\}$ playing the role of "teacher forcing" (Lamb et al., 2016).[2] In both the encoder and the decoder, the hidden states are obtained sequentially by recursively applying f^h and f^s. The final output of interest $\{\mathbf{y}_t\}_{t=1}^{T_y}$ is given by some non-linear function $g(\cdot)$ of the corresponding states $\mathbf{s}_t$. Such a recurrent structure usually poses limitations on learning long-range dependencies (Hochreiter, Bengio, Frasconi, Schmidhuber, et al., 2001) and is subject to the constraints of sequential computation. As it can be seen next, a transformer architecture effectively conducts the same type of mapping as in (1) and (2), but circumvents sequential processing.

[2] Teacher forcing is a strategy of using ground truth values as input during model training. During testing, predicted values from the previous time step of the decoder are used as input.

2.2. Transformer encoder–decoder

A transformer architecture (Vaswani et al., 2017) aims to reduce sequential computation and enable parallelization. To obtain encoder and decoder hidden states, an attention mechanism is used, with which inputs are directly attended. Crucially, it does not rely on any recurrent structure.

Scaled dot-product attention is an important mechanism in the transformer architecture. Attention blocks attend in a parallel manner to all values in the sequence or sequences under consideration, with a query Q matrix and a key–value pair of (K, V) matrices as inputs. The scaled dot-product operator returns an output that has the same dimension as V. That is,

$$\mathrm{ATTN}(Q, K, V) := \mathrm{softmax}\Big(\frac{QK^\top}{\sqrt{d'}}\Big)V; \quad Q, K \in \mathbb{R}^{T\times d'}, V \in \mathbb{R}^{T\times d}. \tag{3}$$

The output is effectively a weighted combination of the values in V, with the weights reflecting a similarity score between queries and keys after scaling. Depending on how Q, K, and V are constructed, the usage of such a mechanism can be further dichotomized into self-attention (or intra-attention) and inter-attention: the former takes one single (multivariate) sequence as the raw input and Q, K, and V are obtained as its linear projections; in the latter case, Q is obtained through the linear projection of one sequence, whereas K and V are obtained through the linear projections of another. In the transformer architecture, the encoder and the decoder networks leverage the intra- and inter-attention mechanisms to construct the mappings in (1) and (2), whose details are given next.

Encoder. The encoder input sequence $\mathbf{x}_t, t = 1, \ldots, T_x$ first goes through a pre-processing step, where positional encoding (Vaswani et al., 2017) is added to the original input sequence. The latter describes the location of an observation in the sequence. We denote the positionally encoded sequence by $\tilde{\mathbf{x}}_t$. The mapping from $\tilde{\mathbf{x}}_t$ to its hidden representation (see also (1)) uses a stack of encoder layers, each comprising the following two sub-blocks: (i) a self-attention block that first (linearly) projects the input to Q, K, V, followed by the scaled dot-product attention operator; and (ii) a feed-forward block that contains a residual connection, layer normalization, and linear layers. Note that the input to the first encoder layer is $\{\tilde{\mathbf{x}}_t\}$, whereas those to subsequent layers are the output from their previous layers. The final encoder output $\{\mathbf{h}_t\}_{t=1}^{T_x}$ is given by the output of the last encoder layer.

Decoder. Analogously to the encoder, the decoder conducts the mapping in (2) through a stack of decoder layers. It takes $\mathbf{y}_t, t = 0, \ldots, T_y - 1$ as well as $\{\mathbf{h}_t\}_{t=1}^{T_x}$ as decoder inputs, with $\{\mathbf{y}_t\}_{t=0}^{T_y-1}$ playing the role of teacher forcing. Each decoder layer (compared to the encoder layer) has an additional encoder–decoder attention block that consumes $\{\mathbf{h}_t\}$. Concretely, it has three sub-blocks: (i) a self-attention (intra-) block similar to the encoder one; (ii) an encoder–decoder (inter-) attention block that projects the output from block (i) into Q and $\{\mathbf{h}_t\}$ into K, V, followed by the scaled dot-product operator; and (iii) a feed-forward block similar to the one in the encoder layer. The input to the first decoder layer is the positionally encoded decoder input $\{\tilde{\mathbf{y}}_t\}_{t=0}^{T_y-1}$, and that to subsequent layers is the output from previous layers.

We highlight some additional nuances in the actual implementation, with more details included in Appendix D:

– The use of a mask. Self-attention in its generic form attends to all values in the input sequence, which for the decoder would run into the risk of "attending to the future". To that end, for each input indexed by t and its corresponding encoded counterpart from the output of the attention operator, masks are applied to inputs with indices $j > t$, to ensure that the output from the attention operator does not depend on values in the future.
– Multi-head attention. Instead of using a single attention operator, multiple attention functions are applied in parallel, so that the model attends to information from representations obtained from different projections (Michel, Levy, & Neubig, 2019; Vaswani et al., 2017).

3. Proposed modeling framework

An encoder–decoder structure is naturally suited for MF data prediction tasks in that (1) it does not require the input and the output to have the same length, which conveniently tackles a key challenge in modeling MF data; and (2) attention mechanisms, irrespective of whether they attend in a temporal (e.g., LSTM) or a parallel manner (e.g., transformer), process and aggregate input information and make it accessible to all targets/outputs, which circumvents the need for frequency alignment that has been a core requirement for a number of existing approaches reviewed in Section 1.1.

To this end, we introduce our proposed multi-task shared-encoder-dual-decoder framework to tackle MF data prediction tasks. The framework consists of a shared encoder that encodes past HF input information, and two decoders that handle prediction tasks for the HF and the LF blocks. Depending on the exact encoder–decoder architecture adopted, the framework can be further dichotomized into either an LSTM- or a transformer-based one. Through joint training of the encoder and the decoders based on minimizing the sum of the loss functions of the two tasks, the proposed framework learns the high-frequency intra-block dynamics that enable the block's r-step-ahead forecasts (task 1), as well as the inter-block dynamics which aid in the low-frequency block's prediction (task 2).

Formally, let the low-frequency block $\{\mathbf{y}\}$ evolve at a resolution of t and the high-frequency block $\{\mathbf{x}\}$ at rt. By definition, $\mathbf{y}_t$ and $\mathbf{x}_{rt}$ are LF and HF observations, respectively, of the same physical timestamp, $\forall t$. For example, in the case where the two blocks correspond to monthly and quarterly ($r = 3$) time series, respectively, they correspond to observations in {Mar, Jun, Sept, Dec}. The problem of interest focuses on the following two conditional probabilities, whose conditional expectations are directly modeled by the proposed framework:

$$\mathbb{P}\Big(\mathbf{x}_{r(t+1)}, \dots, \mathbf{x}_{rt+1} | \mathbf{x}_{rt}, \dots, \mathbf{x}_{rt-T_x+1}\Big) \quad \text{and}$$
$$\mathbb{P}\Big(\mathbf{y}_t | \mathbf{y}_{t-1}, \dots, \mathbf{y}_{t-T_y+1}, \mathbf{x}_{rt}, \dots, \mathbf{x}_{rt-T_x+1}\Big),$$

that is, the forecast of the HF block up to r steps ahead given its own past (up to T_x time lags), and the last-vintage nowcast of the LF block given its own past (up to T_y time lags) as well as that of the HF block. Note that the framework is capable of producing all nowcast vintages as well as multi-horizon forecasts of the LF block, with the aid of the plug-in estimates of the HF one, despite the fact that the quantity being modeled is the last nowcast vintage of the LF block that depends on the HF block's contemporaneous value (see Section 3.4).

Under the proposed shared-encoder-dual-decoder framework, the conditional predictions are obtained via the following sets of mappings, where the (shared encoder) maps the history of the HF block $\{\mathbf{x}_{rt-T_x+1}, \dots, \mathbf{x}_{rt}\}$ to a sequence of intermediate hidden states of the same length. That is,

$$\Phi : \big\{\mathbf{x}_{rt-T_x+1}, \dots, \mathbf{x}_{rt}\big\} \mapsto \big\{\mathbf{h}_1, \dots, \mathbf{h}_{T_x}\big\}. \tag{shared encoder}$$

The two decoders take the encoder hidden states and their respective decoder inputs, map them to their corresponding decoder hidden states, and produce an output based on the latter. That is,

$$\begin{aligned}&\Psi^x : \big\{\mathbf{x}_{rt}, \dots, \mathbf{x}_{rt+r-1}, \mathbf{h}_1, \dots, \mathbf{h}_{T_x}\big\} \mapsto \big\{\mathbf{s}_1^x, \dots, \mathbf{s}_r^x\big\},\\ &g^x : \big\{\mathbf{s}_1^x, \dots, \mathbf{s}_r^x\big\} \mapsto \big\{\mathbf{x}_{rt+1}, \dots, \mathbf{x}_{rt+r}\big\};\end{aligned} \tag{high-frequency decoder}$$

and

$$\begin{aligned}&\Psi^y : \big\{\mathbf{y}_{t-T_y}, \dots, \mathbf{y}_{t-1}, \mathbf{h}_1, \dots, \mathbf{h}_{T_x}\big\} \mapsto \big\{\mathbf{s}_1^y, \dots, \mathbf{s}_{T_y}^y\big\},\\ &g^y : \big\{\mathbf{s}_{T_y}^y\big\} \mapsto \big\{\mathbf{y}_t\big\}.\end{aligned} \tag{low-frequency decoder}$$

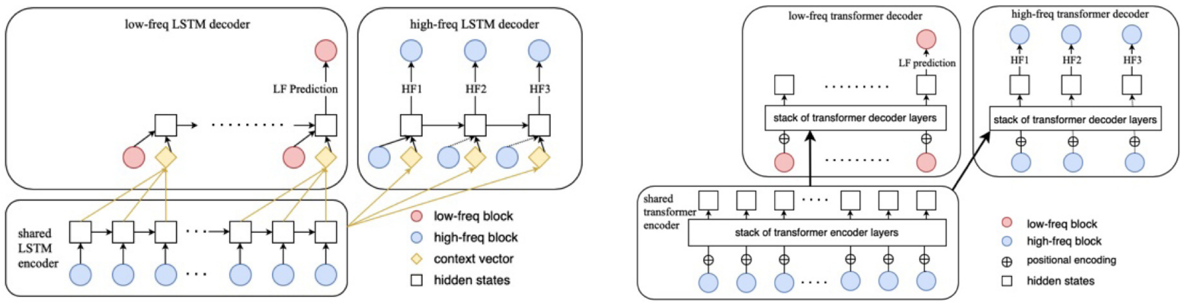
A summary of the steps for model training is given in Exhibit 1.

Next, we expand on the details related to the mappings Φ and Ψ^x, Ψ^y for the LSTM and transformer-based architectures. In the remainder of this section, to convey the key ideas and for ease of exposition, we focus on the case where $T_x = rT_y$, i.e., where the lags for the high- and low-frequency blocks span identical historical periods.

3.1. LSTM-based architecture with temporal local attention

With an LSTM-based architecture (seq2seq), the hidden states are obtained sequentially in both the encoder and the decoders; see the diagram in Fig. 1(a) for an illustration.

Shared high-frequency encoder. Within the encoder mapping Φ, the intermediate hidden states $\{\mathbf{h}_i\}_{i=1}^{T_x}$ are obtained by recursively applying some nonlinear function



(a) an LSTM-based architecture (b) a transformer-based architecture

Fig. 1. Diagram of the proposed framework, with frequency ratio $r = 3$. Dotted arrows (in the high-frequency decoder) correspond to "teacher forcing" during training.

Exhibit 1: Outline of steps for training the multi-task encoder-dual-decoder models.

```
Input: Encoder input $\{\mathbf{x}_{rt-T_x+1}, \cdots, \mathbf{x}_{rt}\}$; HF-decoder input
       $\{\mathbf{x}_{rt}, \cdots, \mathbf{x}_{rt+r-1}\}$ and target $\{\mathbf{x}_{rt+1}, \cdots, \mathbf{x}_{rt+r}\}$;
       LF-decoder input $\mathbf{y}_{t-T_y}, \cdots, \mathbf{y}_{t-1}$ and target $\mathbf{y}_t$
– Forward pass:
    0. Positional encoding (optional for LSTM-based encoder);
    1. Encoding (mapping $\Phi$): pass $\{\mathbf{x}_{rt-T_x+1}, \cdots, \mathbf{x}_{rt}\}$
       through the encoder to obtain hidden states $\{\mathbf{h}_i\}_{i=1}^{T_x}$;
    2. High-frequency decoding (mappings $\Psi^x$ and $g^x$):
       pass hidden states $\{\mathbf{h}_i\}_{i=1}^{T_x}$ and $\{\mathbf{x}_{rt}, \cdots, \mathbf{x}_{rt+r-1}\}$
       through the HF decoder to get $\{\widehat{\mathbf{x}}_{rt+1}, \cdots, \widehat{\mathbf{x}}_{rt+r}\}$;
    3. Low-frequency decoding (mappings $\Psi^y$ and $g^y$): pass
       hidden states $\{\mathbf{h}_i\}_{i=1}^{T_x}$ and $\mathbf{y}_{t-T_y}, \cdots, \mathbf{y}_{t-1}\}$ through the
       LF decoder to get $\widehat{\mathbf{y}}_t$;
    4. Calculate loss based on
       $\Delta(\{\widehat{\mathbf{x}}_{rt+1}, \cdots, \widehat{\mathbf{x}}_{rt+r}\}, \{\mathbf{x}_{rt+1}, \cdots, \mathbf{x}_{rt+r}\})$ and $\Delta(\widehat{\mathbf{y}}_t, \mathbf{y}_t)$
– Backward pass: Update weights based on gradients
  (back-propagation)
Output: Trained encoder mapping $\Phi$, HF decoder
        mappings $\Psi^x, g^x$, and LF decoder mappings $\Psi^y, g^y$
```

f^h on the corresponding encoder input and the previous hidden state. That is, for $i = 1, \ldots, T_x$,

$$\mathbf{h}_i = f^h\left(\mathbf{x}_{rt-T_x+i}, \mathbf{h}_{i-1}\right),$$

for some $\mathbf{h}_0$ of commensurate dimension with either random or zero initialization. Note that f^h is invariant across time steps.

High-frequency decoder. To obtain forecasts of the time series in the HF block, the mapping Ψ^x for the high-frequency decoder entails the recursive application of some function $f^{s,x}$:

$$\begin{aligned} \mathbf{s}_j^x &= f^{s,x}(\mathbf{s}_{j-1}^x, \mathbf{x}_{rt+j-1}, \mathbf{c}_j^x), \\ \mathbf{c}_j^x &:= \sum_{i=1}^{T_x} \alpha_{ji}^x \mathbf{h}_i \end{aligned} \qquad \text{for } j = 1, \ldots, r,$$

where $\mathbf{s}_0^x = \mathbf{h}_{T_x}$, and the weights α_{ji}^x for the context vector $\mathbf{c}_j^x$ are learned through some alignment model (see also Section 2.1). The forecasts are obtained as a function of their corresponding decoder hidden states. That is,

$$\mathbf{x}_{rt+j} = g^x(\mathbf{s}_j^x), \quad j = 1, \ldots, r. \tag{4}$$

Low-frequency decoder. The low-frequency decoder Ψ^y is structured analogously to the high-frequency one. Specifically, it recursively uses some nonlinear function $f^{s,y}$ to obtain the hidden states $\mathbf{s}^y$:

$$\begin{aligned} \mathbf{s}_j^y &= f^{s,y}(\mathbf{s}_{j-1}^y, \mathbf{y}_{(t-1)+j-T_y}, \mathbf{c}_j^y), \\ \mathbf{c}_j^y &:= \sum_{i=r(j-1)+1}^{rj} \alpha_{ji}^y \mathbf{h}_i, \end{aligned} \qquad \text{for } j = 1, \ldots, T_y$$

with $\mathbf{s}_0^y = \mathbf{h}_{T_x}$. Note that the context $\mathbf{c}_j^y$ attends *locally* to the hidden states of the encoder, with the attention window restricted to those that are in the same period as the one corresponding to the LF block. Finally, through some function g^y on the terminal decoder hidden state, the prediction for the low-frequency block is obtained:

$$\mathbf{y}_t = g^y(\mathbf{s}_{T_y}^y). \tag{5}$$

3.2. Transformer-based architecture

In this architecture, the hidden states are obtained in a parallel fashion for both the encoder and the decoders; see the diagram in Fig. 1(b) for an illustration.

Shared high-frequency encoder. The encoder mapping Φ maps the high-frequency input $\{\mathbf{x}\}$ to the collection of hidden states $\{\mathbf{h}\}$ of the same length:

$$\{\mathbf{h}_1, \ldots, \mathbf{h}_{T_x}\} = \Phi\Big(\{\mathbf{x}_{rt-T_x+1}, \ldots, \mathbf{x}_{rt}\}\Big). \tag{6}$$

Note that the LSTM encoder counterpart of Φ entails the recursive application of function f^h that acts on the previous hidden state and the current input. In contrast, the transformer-based encoder mapping obtains the hidden states $\{\mathbf{h}\}$ through parallel processing, and architecturally comprises a stack of encoder layers, as described in Section 2.2, with the latter including a self-attention block and additional layers such as residual or feed-forward ones.

High-frequency decoder. To obtain forecasts of the HF block, the corresponding decoder mapping Ψ^x maps the encoded hidden states $\mathbf{h}$ together with the decoder input sequence $\mathbf{x}_j, j = rt, \ldots, rt + r - 1$ to the decoder hidden states $\{\mathbf{s}^x\}$ as follows:

$$\{\mathbf{s}^x_1, \ldots, \mathbf{s}^x_r\} = \Psi^x\Big(\{\mathbf{x}_j\}_{j=rt}^{rt+r-1}, \{\mathbf{h}_i\}_{i=1}^{T_x};\ \mathbb{M}^x\Big),$$

where $\mathbb{M}^x$ is a look-ahead mask operator, whose role is to prevent Ψ^x from having access to future values of the input sequence $\mathbf{x}_{j'}$, $j' > j$ for decoding steps $j = 1, \ldots, r$ (see the discussion in Section 2.2 and further technical details in Appendix D). The constituent components of Ψ^x encompass a stack of decoder layers, as described in Section 2.2, wherein key modules of each layer include a self-attention block and an encoder–decoder attention block. Subsequently, forecasts of the high-frequency variables are obtained identically to those in an LSTM-based architecture, namely (4).

Low-frequency decoder. The structure of the LF decoder mapping Ψ^y is analogous to that of the HF one:

$$\{\mathbf{s}^y_1, \ldots, \mathbf{s}^y_{T_y}\} = \Psi^y\Big(\{\mathbf{y}_j\}_{j=(t-T_y)}^{t-1}, \{\mathbf{h}_i\}_{i=1}^{T_x};\ \mathbb{M}^y\Big),$$

with $\mathbb{M}^y$ again being a look-ahead mask operator fulfilling the same role as in the HF decoder. The major constituents of the mapping Ψ^y are the same as those in Ψ^x; namely, self-attention and encoder–decoder blocks, etc. Finally, the predicted values of the LF block are obtained through some nonlinear function g^y, as in (5).

3.3. Model training

As noted above, even though the exact computing modules differ depending on whether one chooses an LSTM or a transformer-based architecture, the proposed framework is unified in its structure with a shared encoder governed by mapping Φ, and two decoders that are respectively governed by Ψ^x, g^x and Ψ^y, g^y for the tasks of predicting high- and low-frequency data. Consequently, training these two architectures follows the same steps, as briefly outlined next.

Let ϕ, ψ^x, and ψ^y denote the parameters of the mappings Φ, Ψ^x, and Ψ^y, respectively, and let θ^x and θ^y denote those of g^x and g^y, respectively, as described in Sections 3.1 and 3.2. These model parameters are estimated by minimizing the empirical risk measured by the deviation between the observed values of the time series and those predicted by the corresponding learned mappings.

Concretely, there are N_x and N_y available observations for the high- and low-frequency blocks of time series, respectively, with N_x and N_y satisfying $rN_y \le N_x \le (r+1)N_y$. The observed sequences are arranged into a total number of $N := \min\{\lfloor \frac{N_x - r - T_x}{r} \rfloor, N_y - T_y\}$ samples. Specifically, the nth training sample ($n = 0, \ldots, N-1$) has $\mathcal{X}^{(n)}_{\text{enc}}$ as input to the encoder, and $\mathcal{X}^{(n)}_{\text{dec}}$ and $\mathcal{Y}^{(n)}_{\text{dec}}$ as inputs to the high- and low-frequency decoders, respectively:

$$\mathcal{X}^{(n)}_{\text{enc}} := \{\mathbf{x}_i\}_{i=rn+1}^{rn+T_x}, \quad \mathcal{X}^{(n)}_{\text{dec}} := \{\mathbf{x}_j\}_{j=rn+T_x}^{rn+T_x+r-1}, \quad \mathcal{Y}^{(n)}_{\text{dec}} := \{\mathbf{y}_j\}_{j=n}^{n+T_y-1}.$$

The targets for the two tasks are given by

$$\widetilde{\mathcal{X}}^{(n)} := \{\mathbf{x}_j\}_{j=rn+T_x+1}^{rn+T_x+r} \quad \text{and} \quad \widetilde{\mathcal{Y}}^{(n)} := \{\mathbf{y}_{n+T_y}\}.$$

Estimates of the parameters are obtained by minimizing the following loss function:

$$(\widehat{\phi}, \widehat{\psi}^x, \widehat{\psi}^y, \widehat{\theta}^x, \widehat{\theta}^y) = \operatorname{argmin} \mathcal{L}(\phi, \psi, \omega)$$
$$:= \frac{1}{N}\sum_{n=0}^{N-1} \ell^{(n)}\Big(\phi, \psi^x, \psi^y, \theta^x, \theta^y; \mathcal{X}^{(n)}_{\text{enc}}, \mathcal{X}^{(n)}_{\text{dec}}, \mathcal{Y}^{(n)}_{\text{dec}}, \widetilde{\mathcal{X}}^{(n)}, \widetilde{\mathcal{Y}}^{(n)}\Big),$$

where $\ell^{(n)}$ denotes the loss for the nth training sample and in the case of the ℓ_2 loss is given by:

$$\ell^{(n)}(\phi, \psi^x, \psi^y, \theta^x, \theta^y) = w_0 \Big\|\widehat{\mathbf{y}}_{n+T_y} - \mathbf{y}_{n+T_y}\Big\|_2^2 + \sum_{j=1}^{r} w_j \Big\|\widehat{\mathbf{x}}_{rn+T_x+j} - \mathbf{x}_{rn+T_x+j}\Big\|_2^2,$$

for positive weights satisfying $\sum_{j=0}^r w_j = 1$. $\widehat{\mathbf{x}}_{rn+T_x+j}, j = 1, \ldots, r$ and $\widehat{\mathbf{y}}_{n+T_y}$ are obtained by having the learned mappings acting on the inputs $\mathcal{X}^{(n)}_{\text{enc}}, \mathcal{X}^{(n)}_{\text{dec}}, \mathcal{Y}^{(n)}_{\text{dec}}$ and the corresponding intermediate estimates.

3.4. Forecasting and nowcasting

Based on the learned mappings $\Phi, \Psi^x, \Psi^y, g^x, g^y$ and available observations for the high- and the low-frequency blocks up to timestamps N_x and N_y (recall that $rN_y \le N_x \le (r+1)N_y$), we can make forecasts/nowcasts for both blocks. In what follows, we use a hat ($\widehat{\cdot}$) to denote the model's predicted values to distinguish them from observed ones, and we elaborate on how forecasts and nowcasts can be obtained based on the trained model.

Single-horizon forecast. For forecasts, note that $N_x = rN_y$; that is, both blocks have observations up to the same physical timestamp. The predictions of interest are $\mathbf{x}_{N_x+1}, \ldots, \mathbf{x}_{N_x+r}$ and $\mathbf{y}_{N_y+1}$, i.e., the values for the HF and LF blocks in the next period. They can be obtained by first producing forecasts for the HF block and then using them as plug-in estimates to produce that of the LF one.

Specifically, the encoder hidden states are given by

$$\big\{\widehat{\mathbf{h}}_1, \ldots, \widehat{\mathbf{h}}_{T_x}\big\} = \Phi\Big(\big\{\mathbf{x}_{rN_y-T_x+1}, \ldots, \mathbf{x}_{rN_y}\big\};\ \widehat{\phi}\Big). \tag{7}$$

For r-step-ahead forecasts of the HF block, with a slight abuse of notation, its decoder hidden states and the corresponding forecasts can be recursively obtained through

$$\begin{aligned} \widehat{\mathbf{s}}^x_j &= \Phi^x\Big(\{\widehat{\mathbf{h}}_i\}_{i=1}^{T_x}, \{\widehat{\mathbf{x}}_{N_x+\ell}\}_{\ell=0}^{j-1};\ \widehat{\psi}^x\Big) \quad \text{and} \\ \widehat{\mathbf{x}}_{N_x+j} &= g^x\big(\widehat{\mathbf{s}}^x_j;\ \widehat{\theta}^x\big); \qquad \text{for } j = 1, \ldots, r. \end{aligned} \tag{8}$$

Note that the recursion in the LSTM and the look-ahead mask in the transformer enable both architectures (and thus Φ^x) to sequentially obtain the decoder hidden states without relying on the "future" decoder input. With the estimated $\{\widehat{\mathbf{x}}_{N_x+j}\}_{j=1}^r$, we re-encode to obtain LF block forecasts. Concretely, let

$$\big\{\widetilde{\mathbf{h}}_1, \ldots, \widetilde{\mathbf{h}}_{T_x}\big\} = \Phi\ \Big(\Big\{\ \mathbf{x}_{rN_y-T_x+1+r}, \ldots, \mathbf{x}_{rN_y}, \widehat{\mathbf{x}}_{rN_y+1}, \ldots, \widehat{\mathbf{x}}_{rN_y+r}\ \Big\};\ \widehat{\phi}\ \Big). \tag{9}$$

That is, we apply the encoder mapping to obtain re-encoded hidden states based on the last $T_x - r$ actually observed values and the r estimated ones of the HF block. The LF decoder hidden states and the low-frequency forecast are then obtained by

$$\left\{\widehat{\mathbf{s}}^y_1, \ldots, \widehat{\mathbf{s}}^y_{T_y}\right\} = \Psi^y\left(\{\widetilde{\mathbf{h}}_i\}_{i=1}^{T_x}, \{\mathbf{y}_\ell\}_{\ell=N_y-T_y+1}^{N_y};\ \widehat{\psi}^y\right) \quad \text{and}$$
$$\widehat{\mathbf{y}}_{N_y+1} = g^y\left(\widehat{\mathbf{s}}^y_{T_y};\ \widehat{\theta}^y\right). \tag{10}$$

Nowcast vintage k, $1 \leq k \leq r$. The kth nowcast vintage, denoted by N_k, corresponds to having observed the actual values of the HF block k steps into the next period, i.e., $N_x = rN_y + k$. The predictions of interest are given by $\mathbf{x}_{N_x+1}, \ldots, \mathbf{x}_{N_x+r-k}$ and $\mathbf{y}_{N_y+1}$, that is, values of the next $(r-k)$ steps of the HF block that remain unobserved and the value of the LF block for the period in question. Note that $N_x + r - k \equiv (r+1)N_y$ by definition. Predictions can be obtained in a similar fashion to the forecast case, with minimal modification to the HF decoding. Specifically, forecasts for the remaining steps in the period of the HF block are obtained as follows:

- Obtain encoder hidden states according to (7).
- Obtain high-frequency forecasts similar to (8), with the decoder input substituted by the observed values wherever applicable. That is, for $j = 1, \ldots, r$,

  $$\widehat{\mathbf{s}}^x_j = \Phi^x\left(\{\widehat{\mathbf{h}}_i\}_{i=1}^{T_x}, \{\widehat{\mathbf{x}}_{rN_y+\ell}\}_{\ell=0}^{j-1};\ \widehat{\psi}^x\right)$$

  with $\widehat{\mathbf{x}}_{rN_y+\ell} \equiv \mathbf{x}_{N_x+\ell-k}$, $\forall\, \ell = 1, \ldots, k$ and the $(r-k)$ predicted values given by

  $$\widehat{\mathbf{x}}_{N_x+\ell} = g^x\left(\widehat{\mathbf{s}}^x_{k+\ell};\ \widehat{\theta}^x\right), \quad \text{for} \quad \ell = 1, \ldots, r-k.$$

The remainder of the process follows. That is, one proceeds by using (9) and (10) to obtain the low-frequency prediction. Note that in the case where $k = r$, all observations of the HF block for the next period are available. One can disregard the steps pertaining to HF block forecasts and directly proceed with an LF block nowcast.

Multi-horizon forecasts. Producing predictions for multiple periods is a direct extension of that for a single horizon. In particular, for H-period-ahead prediction, the values of interest are

$$\left\{\mathbf{y}_{N_y+1}, \ldots, \mathbf{y}_{N_y+H}\right\} \quad \text{and}$$
$$\left\{\mathbf{x}_{N_x+1}, \ldots, \mathbf{x}_{r(N_y+1)}, \ldots, \mathbf{x}_{r(N_y+H)}\right\},$$

for the low- and high-frequency blocks, respectively. One can then proceed as follows:

- For period 1, depending on the number of available observations of the high-frequency block into the period, the predictions $\mathbf{y}_{N_y+1}$ and $\{\mathbf{x}_{N_x+1}, \ldots, \mathbf{x}_{r(N_y+1)}\}$ can be obtained by following the aforementioned steps in either the single-horizon forecast or nowcast case.
- For period $h = 2, \ldots, H$, one repeats the steps in the single-horizon forecasts, while shifting the encoding/decoding windows accordingly, by disregarding observed values at the front and appending estimates at the end; the estimates are the predictions coming from previous periods.

The proposed framework based on the seq2seq and transformer architectures handles forecasting/nowcasting tasks in a unified manner. It posits a single data generating process regarding the respective dynamics within both the high- and low-frequency blocks, as well as the association between them. The difference between forecasts and nowcast vintages is captured by the input sequence used by the HF decoder; namely, predicted values for forecasting tasks and actual values for nowcasting ones. In contrast, many existing regression and vector autoregression models in the literature require a re-specification of their temporal dynamics for handling nowcasting tasks (see also Appendix A).

4. Experiments on synthetic data

We assess the proposed framework in terms of its forecasting/nowcasting performance on different data generating mechanisms, and compare it against several benchmarks that encompass linear, tree-based, and neural network-based models. Depending on the working model adopted, these models largely fall into the following four categories, allowing for nonlinearity:

1. Autoregressive distributed lag (ADL) models, where each low-frequency/high-frequency series is modeled as a univariate response and its dynamics depend on lags of its own and others. Models or engines within this category include a multi-layer perceptron (`mlp`), gradient boosted machine (`gbm`) (Ke et al., 2017), and MIDAS (Ghysels et al., 2007) with different polynomial specifications, that is, Almon (`almon`), beta (`nbeta`), and unrestricted (`unres`).
2. Vector autoregressive (VAR) models, where the low- and high-frequency blocks are modeled as a joint VAR system with frequency alignment performed. In particular, we consider the multivariate extensions of DeepAR (`deepvar`) (Salinas, Flunkert, Gasthaus, & Januschowski, 2020) and NHiTS (`nhits`) (Challu et al., 2022) to handle this system, which requires model re-specification across different vintages.
3. The mixed-frequency Bayesian VAR (`mfbvar`) model (Schorfheide & Song, 2015).
4. Univariate time series models where the interdependency across coordinates is ignored. Such models include ARIMA (`arima`) and the simple exponential smoother (`ses`) (e.g., Makridakis et al., 2008), with the latter being a special case that falls under the state-space framework (Hyndman, Koehler, Snyder, & Grose, 2002).

Within our proposed multi-task framework, we also consider a sequence-to-one (`seq2one`) architecture variant, where the high-frequency decoder has a simplified output head that produces only one-step-ahead forecasts instead of multi-steps for the entire period.

Additional details on the specification of these classes of competing models are given in Appendix A.

4.1. Experimental settings

Synthetic data are generated according to two sets of models, the first based on state-space models and the other on regression models. The frequency ratio is set to $r = 3$ across all settings, partly to accommodate the available implementation of some of the competing models. These two data generating mechanisms are presented below, with the exact equation for each setting given in Appendix B.

State-space models. The dynamics of the latent state process $\mathbf{f}_t$ are given by a linear VAR(2) model as follows:

$$\mathbf{f}_t = \Gamma_1 \mathbf{f}_{t-1} + \Gamma_2 \mathbf{f}_{t-2} + \mathbf{e}^f_t, \quad \text{where } \mathbf{e}^f_t \sim \mathcal{N}(0,1).$$

Both the high- and low-frequency blocks are generated according to a VAR(1)-X model and indexed by t for the former and by t' for the latter (note that t' and rt correspond to the same "physical" timestamp):

$$\mathbf{x}_t = A_1 \mathbf{x}_{t-1} + \sum_{i=0}^{q_{fx}} \Lambda^{fx}_i \pi_i(\mathbf{f}_{t-i}) + \mathbf{e}^x_t,$$
$$\text{where } \mathbf{e}^x_t \sim \tfrac{1}{\sqrt{6}} \cdot \mathbf{t}_8, \quad (t\text{-distribution with df} = 8)$$
$$\mathbf{y}_{t'} = B_1 \mathbf{y}_{t'-1} + \sum_{i=0}^{q_{fy}} \Lambda^{fy}_i \varrho_i(\mathbf{f}_{rt-i}) + \mathbf{e}^y_{t'},$$
$$\text{where } \mathbf{e}^y_{t'} \sim \mathcal{N}(0,1).$$

The autoregressive dynamics for both blocks are linear, and all noise processes have independent coordinates with unit variance. Further, $\pi_i(\cdot)$ and $\varrho_i(\cdot)$ are either identity or radial-basis functions (rbf), with the latter introducing nonlinear dependency on $\mathbf{f}_t$. In particular, to represent rbfs we consider radial-basis function networks that have one hidden layer and the same number of output neurons as the input ones. Additionally, the output neurons are scaled so that they have the same standard deviation as the input ones.

All entries of the autoregressive coefficient matrices $\Gamma_1, \Gamma_2, A_1, B_1$ are initially generated from $\mathsf{Unif}\{(-2,-1) \cup (1,2)\}$, then scaled so that the spectral radius of the corresponding companion matrix is smaller than 1 to ensure the stability of the process. For each entry (j,k) in Λ^{fx}, $\{\Lambda^{fx}_i(j,k); i = 0,\ldots,q_{fx}\}$ is generated using an Almon polynomial of degree 2, so that the sequence over i is decreasing in magnitude. In practice, we also randomly flip the sign of the generated sequence so that the coefficients can also be negative. The entries for Λ^{fy} are generated in an identical manner. Given the decreasing-in-magnitude nature of Λ_i, the degree of nonlinearity of a data generating process (DGP) can be controlled by the exact specification of π_i and ϱ_i. Concretely, in the case where all the π_i and ϱ_i are identity functions, the DGP is linear. When all the π_i and ϱ_i are rbfs, the DGP becomes highly nonlinear. In the case where π_i and ρ_i are identity functions for $i \le i'$ and rbfs for $i > i'$, the DGP exhibits a varied degree of nonlinearity.

Regression models. The dynamics of $\mathbf{x}_t$ are given by a nonlinear VAR(2) model:

$$\mathbf{x}_t = A_1 \mathbf{x}_{t-1} + A_2 \Big(\mathbf{x}_{t-2} \circ \sin\big(\mathbf{x}_{t-2}\big)\Big) + \mathbf{e}^x_t,$$
$$\text{where } \mathbf{e}^x_t \sim \tfrac{2}{\sqrt{6}} \cdot \mathbf{t}_8,$$

where the sin operation is applied in an element-wise manner ($\circ$ denotes element-wise multiplication). The dynamics of the low-frequency block are similar to its observation equation in the state-space model case, i.e.,

$$\mathbf{y}_{t'} = B_1 \mathbf{y}_{t'-1} + \sum_{i=0}^{q_{xy}} \Lambda^{xy}_i \varrho_i(\mathbf{x}_{rt-i}) + \mathbf{e}^y_{t'}$$
$$\text{where } \mathbf{e}^y_t \sim \mathcal{N}(0, 1/4).$$

Once again, A_1, A_2, B_1 are initially generated from some uniform distribution and then scaled to satisfy the spectral radius $<$ 1 constraint; and Λ^{xy}_i and ϱ_i are treated analogously to the case of state-space models. Table 1 outlines all settings considered.

To train the models, we consider training sample sizes of {500, 1000, 3000}, with $T_x \equiv 12$ and $T_y \equiv 4$ for all settings. For all tree- and neural network-based models, we reserve a validation set of size 200 and conduct hyper-parameter tuning using Bayesian optimization, based on the validation set performance for a training sample size of 1000.[3] The selected hyper-parameters are then also used for training sample sizes of 500 and 3000. In particular, for `gbm`, we use LightGBM (Ke et al., 2017); for `deepar` and `nhits`, we leverage the implementation and API in the `Darts Python` library (Herzen et al., 2022); and for MIDAS and mfbVAR, we leverage the implementation in `R` packages `midasr` (Ghysels, Kvedaras, & Zemlys, 2016) and `mfbvar` (Ankargren & Yang, 2021).[4]

Once the models are trained, we evaluate their performance on both forecasting and nowcasting tasks over a test set whose size is fixed at 100. For each test sample $\tau = 1,\ldots,100$, we conduct forecast F as well as nowcasts $\mathsf{N}_1, \mathsf{N}_2, \mathsf{N}_3$.[5] And we predict up to H periods ahead with $H \equiv 4$, i.e., for each set of inputs in a test sample, for vintage N_k, $k \in \{0,1,2,3\}$ (with $\mathsf{F} \equiv \mathsf{N}_0$). Predicted values are given by $\{\widehat{\mathbf{x}}_{k+1},\ldots,\widehat{\mathbf{x}}_{12}\}$ and $\{\widehat{\mathbf{y}}_1,\ldots,\widehat{\mathbf{y}}_4\}$, where subscripts correspond to the relative steps to the last available data point of the forecasting case (see the additional illustration in Table B.7). Note that for the MIDAS

[3] See Appendix B.2 for additional details on hyper-parameter selection.

[4] In `midasr`, for the Almon polynomial (`almon`), its degree is fixed at 2 and the initial values are set as [1, −0.5]. For the beta polynomial (`nbeta`), its degree is fixed at 3 and the initial values are set as [0.5, −0.5, 0.5]. For `mfbvar`, we use the Minnesota prior distribution with the default parameters provided in the package.

[5] See Section 3.4 for the definition of the general case, where N_k corresponds to the Nowcast vintage where there are $\underline{k}$ additionally available data points "into" the period of interest for the high-frequency block. For the experiments in questions, given that the frequency ratio is set at 3, the high-frequency block can have at most three such (additional) data points available. As a concrete example, in the case of monthly–quarterly variables where low-frequency data are available up to December, N_1 refers to the nowcast where high-frequency data corresponding to January of the next calendar year are available and used in the prediction; $\mathsf{N}_2, \mathsf{N}_3$ are analogously defined.

Table 1
Configuration for simulations. "stsp" corresponds to settings where the DGP is a state-space model, "regr" corresponds to settings where the DGP is a regression model.

	$\dim_f$	d_x	d_y	q_{fx}	q_{fy}	q_{xy}	Remarks
stsp00	5	50	10	15	6	–	linear; π_i's and ϱ_i's are all identity functions
stsp01	5	50	10	12	6	–	mildly nonlinear
stsp02	5	100	10	6	6	–	highly nonlinear; π_i's and ϱ_i's are all rbfs
regr01	–	50	10	–	–	6	mildly nonlinear
regr02	–	100	20	–	–	6	highly nonlinear; ϱ_i's are all rbfs

regression model, the true values of the high-frequency block are provided, since the model does not generate such forecasts.

4.2. Performance evaluation

For each test indexed by $\tau = 1, \dots, 100$, the raw error is calculated by first obtaining the coordinate-wise mean absolute percentage error (MAPE) and then taking the median, for each step j into the forecast horizon:

$$\text{err-x}_j^{\tau,[k]} := \underset{s\in\{1,\dots,d_x\}}{\text{median}}\left\{\left|\frac{\widehat{\mathbf{x}}_{j,s}^{\tau,[k]} - \mathbf{x}_{j,s}^{\tau}}{\mathbf{x}_{j,s}^{\tau}}\right|\right\} \quad \text{and}$$
$$\text{err-y}_j^{\tau,[k]} := \underset{s\in\{1,\dots,d_y\}}{\text{median}}\left\{\left|\frac{\widehat{\mathbf{y}}_{j,s}^{\tau,[k]} - \mathbf{y}_{j,s}^{\tau}}{\mathbf{y}_{j,s}^{\tau}}\right|\right\}, \tag{11}$$

where the superscript $[k]$ indexes the vintages and the subscript s the coordinates. The "summary" error of step j is obtained by taking the median of err-$\text{x}_j^{\tau,[k]}$ and err-$\text{y}_j^{\tau,[k]}$ across all 100 tests. Finally, for all models, we normalize the summary error by that of the `ses`, whose the smoothing parameter is fixed at 0.5. One would expect the model to have similar performance to a simple exponential smoother if the normalized error is close to 1, and to outperform by a wide margin if it is close to 0.

Tables 2 and 3 show the performance evaluation for $\widehat{\mathbf{y}}_1$ and $\widehat{\mathbf{x}}_4$, respectively,[6] with the reported numbers corresponding to the metrics in (11) for the proposed framework and the aforementioned competitor models, after normalizing by that of `ses`.[7] In particular, `seq2seq` and `transf` correspond to the LSTM and transformer-based architectures described in the main text, and `seq2one` is a variant of the former with a simplified output head. Other model codes correspond to the various benchmarks described at the beginning of this section. Finally, the `naive` forecast method, that is, $\widehat{\mathbf{x}}_{T+h} \equiv \mathbf{x}_T$ ($\widehat{\mathbf{y}}_{T+h}$ is analogously defined), is included for validation purposes and is expected to perform slightly worse than `ses` in general.

The major observations based on the tabulated results are three-fold. First, the performance for the developed multi-task models, `mfbvar` and `gbm`, improves across vintages $(\mathsf{F}, \mathsf{N}_1, \mathsf{N}_2, \mathsf{N}_3)$ for both the HF and LF blocks, although the degree of improvement varies across settings. `deepvar` and `nhits` show no noticeable improvement across vintages for the low-frequency block, while the improvement for the high-frequency one is marginal. Note that these two models fall under the MF VAR modeling category, and the observed performance is a consequence of how it handles MF data. See the brief discussion in Remark 3 in Appendix A.2. Further, note that for the MIDAS model, the N_1 and N_2 vintages exhibit worse performance than F. This is likely due to the fact that the constraining schema used for the lag polynomials are different from the true underlying DGP. Second, all competing models broadly exhibit comparable performance on the high-frequency block, whereas the NN-based models are generally more stable and perform better for the low-frequency block. Third, all NN-based models show steady improvement in performance as the training sample size increases. However, such improvement is essentially muted for the autoregressive distributed lag, the MF VAR, and the univariate time series models.

It is worth noting that when the training sample size is small, NN-based models do not exhibit an advantage over competing ones and `mfbvar` in particular, even in the presence of nonlinear dynamics. However, their performance picks up and they typically outperform as more training samples become available. Among the NN-based models, `seq2seq` consistently exhibits the best performance, whereas `transf` is less stable and typically requires a larger sample size to achieve comparable performance. Finally, the performance of the MIDAS regression models (`almon`, `nbeta`, and `unres`) is not particularly competitive against the other models considered, despite the true value of the HF block being provided to the model at the time of prediction.

5. Experiments on US macroeconomic data

We apply the proposed framework to a dataset comprising 23 quarterly and 91 monthly US macroeconomic indicators (see Appendix E) that span the period from January 1960 to March 2022. The data are obtained from the Federal Reserve Economic Data (FRED) database and are transformed according to the recommendations in McCracken and Ng (2016, 2020); see Table E.10 for a full description.

We consider several classes of models, including the developed NN-based multi-task ones, selected ADL ones—that is, `gbm` and `mlp`, which performed well overall in the synthetic data experiments—the existing MIDAS model, and the MF Bayesian VAR model (Schorfheide & Song,

[6] The choice of $\widehat{\mathbf{x}}_4$ is out of consideration that it would also indirectly inform how the error for the HF block would change across different forecasting steps, as the vintages vary.

[7] For the sake of completeness, the absolute metric values in (11) for `ses` are included in Table B.8 in Appendix B Table B.8

Table 2
Performance evaluation for $\widehat{\mathbf{y}}_1$ across different vintages and models.

		500				1000				3000			
		F	N_1	N_2	N_3	F	N_1	N_2	N_3	F	N_1	N_2	N_3
stsp00	seq2seq	0.57	**0.44**	0.36	**0.16**	0.56	0.48	**0.33**	**0.14**	**0.51**	**0.42**	**0.30**	**0.09**
	transf	0.68	0.59	0.56	0.48	0.57	0.54	0.44	0.32	0.56	0.51	0.41	0.26
	seq2one	**0.55**	0.50	**0.35**	0.17	**0.54**	**0.47**	**0.33**	**0.14**	**0.51**	0.43	0.31	0.12
	gbm	0.65	0.65	0.56	0.46	0.63	0.60	0.50	0.42	0.59	0.53	0.48	0.37
	mlp	0.65	0.59	0.50	0.37	0.59	0.54	0.43	0.28	0.55	0.44	0.32	**0.09**
	deepvar	0.83	0.82	0.84	0.80	0.80	0.77	0.76	0.75	0.73	0.72	0.72	0.70
	nhits	0.83	0.80	0.77	0.78	0.75	0.76	0.77	0.74	0.70	0.73	0.72	0.72
	almon	0.61	1.29	1.26	0.37	0.60	1.27	1.22	0.38	0.57	1.16	1.12	0.34
	nbeta	0.88	1.66	1.84	0.79	0.96	1.73	1.81	0.81	0.77	1.56	1.77	0.77
	unres	–	–	–	–	6.10	5.67	3.90	0.46	1.04	1.24	1.02	0.25
	mfbvar	**0.55**	0.52	0.47	0.40	0.55	0.51	0.43	0.34	0.54	0.47	0.44	0.34
	arima	0.82	–	–	–	0.81	–	–	–	0.79	–	–	–
	naive	1.25	–	–	–	1.25	–	–	–	1.25	–	–	–
stsp01	seq2seq	0.72	0.65	**0.55**	0.44	0.68	0.58	0.51	**0.34**	0.61	**0.51**	**0.39**	0.30
	transf	0.75	0.68	0.61	0.58	0.75	0.63	0.52	0.47	**0.60**	0.55	0.49	0.39
	seq2one	0.74	0.67	0.57	0.45	0.70	**0.57**	**0.47**	0.35	0.64	0.53	0.44	0.29
	gbm	0.76	0.75	0.66	0.57	0.70	0.64	0.60	0.51	0.67	0.61	0.55	0.46
	mlp	0.71	0.67	0.60	0.50	0.67	0.66	0.55	0.48	0.68	0.55	0.41	**0.27**
	deepvar	0.81	0.81	0.81	0.80	0.78	0.80	0.75	0.75	0.72	0.73	0.74	0.72
	nhits	0.85	0.83	0.84	0.81	0.84	0.82	0.82	0.80	0.76	0.72	0.78	0.75
	almon	**0.64**	1.23	1.09	**0.43**	0.63	1.07	1.08	0.43	0.65	1.04	1.00	0.49
	nbeta	0.96	1.49	1.48	0.82	0.83	1.55	1.49	0.75	0.84	1.47	1.38	0.80
	unres	–	–	–	–	15.11	17.40	15.09	8.97	2.88	3.59	3.30	1.78
	mfbvar	0.69	**0.60**	0.56	0.48	**0.61**	0.62	0.57	0.51	0.65	0.60	0.56	0.46
	arima	0.78	–	–	–	0.77	–	–	–	0.77	–	–	–
	naive	1.17	–	–	–	1.17	–	–	–	1.17	–	–	–
stsp02	seq2seq	**0.75**	**0.76**	0.80	**0.69**	0.72	**0.71**	**0.66**	**0.59**	**0.72**	**0.66**	**0.64**	**0.55**
	transf	0.79	0.77	0.78	0.78	0.77	0.75	0.76	0.71	**0.72**	0.69	0.70	0.56
	seq2one	0.80	0.78	0.78	0.71	0.79	**0.71**	0.68	0.61	0.75	0.71	0.70	0.59
	gbm	0.79	0.78	0.78	0.74	0.76	0.76	0.73	0.68	**0.72**	0.71	0.72	0.66
	mlp	0.77	**0.76**	**0.77**	0.74	0.78	0.79	0.77	0.71	0.76	0.70	0.70	0.58
	deepvar	0.92	0.90	0.91	0.90	0.87	0.89	0.88	0.87	0.82	0.84	0.82	0.81
	nhits	0.91	0.90	0.88	0.88	0.88	0.87	0.85	0.86	0.83	0.83	0.83	0.80
	almon	1.00	1.12	0.99	0.94	0.80	0.98	0.87	0.73	0.83	0.91	0.89	0.75
	nbeta	1.22	1.30	1.35	1.21	0.99	1.05	1.07	0.96	0.91	1.09	1.01	0.90
	unres	–	–	–	–	–	–	–	–	2.40	3.88	3.60	2.67
	mfbvar	**0.75**	0.79	0.80	0.83	**0.70**	**0.71**	0.77	0.71	0.75	0.74	0.79	0.69
	arima	0.80	–	–	–	0.83	–	–	–	0.82	–	–	–
	naive	1.20	–	–	–	1.20	–	–	–	1.20	–	–	–
regr01	seq2seq	0.84	0.80	0.76	0.66	**0.73**	**0.63**	**0.51**	**0.31**	**0.72**	0.59	**0.50**	**0.19**
	transf	0.87	0.82	0.85	0.84	0.82	0.85	0.83	0.83	0.83	0.80	0.78	0.76
	seq2one	0.85	0.85	0.80	0.78	0.77	0.66	0.57	0.38	**0.72**	**0.58**	**0.50**	0.20
	gbm	0.85	0.84	0.82	0.78	0.84	0.81	0.78	0.74	0.82	0.81	0.76	0.72
	mlp	0.79	0.76	0.72	0.64	0.77	0.73	0.62	0.51	0.74	0.62	0.46	0.29
	deepvar	0.83	0.84	0.83	0.83	0.82	0.79	0.81	0.80	0.84	0.80	0.76	0.76
	nhits	0.85	0.85	0.86	0.87	0.83	0.84	0.84	0.82	0.82	0.83	0.80	0.80
	almon	0.86	0.93	0.93	0.78	0.82	0.99	0.92	0.76	0.78	0.99	0.90	0.73
	nbeta	0.91	1.23	1.37	1.00	0.90	1.30	1.25	0.89	0.92	1.17	1.25	0.87
	unres	–	–	–	–	2.12	1.88	1.56	0.57	0.78	0.93	0.99	0.30
	mfbvar	**0.77**	**0.72**	**0.67**	**0.60**	0.78	0.74	0.65	0.63	0.74	0.69	0.64	0.55
	arima	0.84	–	–	–	0.86	–	–	–	0.86	–	–	–
	naive	1.12	–	–	–	1.12	–	–	–	1.12	–	–	–
regr02	seq2seq	0.89	0.92	0.86	0.82	0.79	0.78	**0.68**	**0.62**	**0.70**	**0.68**	**0.57**	**0.52**
	tranf	0.93	0.90	0.86	0.88	0.82	0.79	0.78	0.80	0.75	0.73	0.73	0.70
	seq2one	0.93	0.92	0.84	0.83	0.87	0.85	0.76	0.74	0.78	0.72	0.59	**0.52**
	gbm	0.80	**0.77**	0.78	0.78	0.77	0.76	0.75	0.74	0.75	0.74	0.72	0.71
	mlp	0.81	0.82	0.79	0.78	0.80	0.75	0.75	0.72	0.72	**0.68**	0.64	0.61
	deepvar	0.85	0.84	0.84	0.86	0.80	0.82	0.81	0.81	0.75	0.74	0.73	0.73
	nhits	0.91	0.89	0.90	0.88	0.84	0.82	0.83	0.83	0.80	0.81	0.78	0.77
	almon	0.94	1.00	1.04	0.87	0.84	0.94	0.93	0.86	0.77	0.86	0.88	0.79
	nbeta	1.12	1.29	1.41	1.22	1.07	1.21	1.22	1.03	0.88	1.02	1.07	0.91
	unres	–	–	–	–	–	–	–	–	1.07	1.09	1.13	0.90
	mfbvar	**0.77**	**0.77**	**0.69**	**0.66**	**0.75**	**0.73**	0.71	0.66	0.74	0.69	0.60	0.60
	arima	0.85	–	–	–	0.84	–	–	–	0.84	–	–	–
	naive	1.20	–	–	–	1.20	–	–	–	1.20	–	–	–

Table 3
Performance evaluation for $\widehat{\mathbf{x}}_4$ across different vintages and models.

		500				1000				3000			
		F	N_1	N_2	N_3	F	N_1	N_2	N_3	F	N_1	N_2	N_3
stsp00	seq2seq	**0.79**	0.72	0.57	0.39	0.76	0.74	0.57	0.37	0.73	**0.69**	**0.50**	**0.34**
	transf	0.80	0.82	0.61	0.42	0.78	0.75	0.58	0.37	0.75	0.72	0.55	0.36
	seq2one	0.82	0.76	0.62	0.39	0.78	0.73	0.52	0.38	**0.71**	0.70	0.52	0.33
	gbm	0.78	0.76	0.69	0.55	0.78	0.77	0.73	0.51	0.76	0.75	0.62	0.48
	mlp	0.81	0.78	0.63	0.49	0.78	0.77	0.62	0.45	0.75	0.75	0.59	0.39
	deepar	0.83	0.82	0.75	0.69	0.85	0.82	0.73	0.63	0.83	0.80	0.67	0.52
	nhits	0.89	0.86	0.79	0.67	0.83	0.82	0.75	0.62	0.81	0.82	0.70	0.58
	mfbvar	0.75	**0.70**	**0.53**	**0.34**	**0.74**	**0.70**	**0.53**	**0.34**	0.75	0.70	0.53	0.35
	arima	0.84	0.83	0.85	0.77	0.84	0.82	0.82	0.78	0.84	0.82	0.83	0.77
	naive	1.29	1.19	1.28	1.27	1.29	1.19	1.28	1.27	1.29	1.19	1.28	1.27
stsp01	seq2seq	0.63	0.55	0.49	0.40	0.64	0.53	0.47	0.36	0.60	0.54	0.45	0.32
	transf	0.68	0.57	0.53	0.40	0.65	0.58	0.49	0.39	0.60	0.52	0.44	0.36
	seq2one	0.64	0.53	0.52	0.46	0.63	0.53	0.49	0.38	0.58	0.54	0.46	0.34
	gbm	0.67	0.59	0.58	0.56	0.63	0.58	0.56	0.52	0.63	0.54	0.52	0.47
	mlp	0.65	0.60	0.56	0.52	0.65	0.56	0.54	0.46	0.60	0.51	0.46	0.34
	deepvar	0.68	0.67	0.66	0.67	0.69	0.65	0.60	0.60	0.71	0.56	0.54	0.47
	nhits	0.73	0.69	0.66	0.66	0.73	0.70	0.65	0.65	0.66	0.65	0.61	0.61
	mfbvar	**0.61**	**0.52**	**0.44**	**0.30**	**0.61**	**0.52**	**0.45**	**0.30**	**0.59**	**0.50**	**0.43**	**0.29**
	arima	0.78	0.75	0.76	0.75	0.77	0.76	0.74	0.74	0.77	0.74	0.74	0.76
	naive	1.12	1.11	1.09	1.07	1.12	1.11	1.09	1.07	1.12	1.11	1.09	1.07
stsp02	seq2seq	0.87	0.79	0.68	0.60	0.82	0.77	0.66	0.59	0.82	0.74	0.62	0.53
	transf	0.88	0.84	0.70	0.63	0.91	0.81	0.69	0.60	0.88	0.77	0.67	0.61
	seq2one	0.91	0.79	0.70	0.65	0.87	0.77	0.66	0.64	0.85	0.75	0.64	0.60
	gbm	**0.82**	0.80	0.73	0.72	**0.80**	0.79	0.67	0.62	**0.80**	0.75	0.64	0.57
	mlp	0.86	0.79	0.70	0.66	0.84	0.79	0.67	0.62	0.84	0.75	0.65	0.58
	deepvar	0.85	0.80	0.73	0.70	0.84	0.78	0.70	0.68	0.83	0.77	0.69	0.64
	nhits	0.88	0.84	0.78	0.75	0.86	0.82	0.76	0.74	**0.80**	0.77	0.72	0.68
	mfbvar	0.83	**0.72**	**0.62**	**0.48**	0.81	**0.73**	**0.57**	**0.50**	0.81	**0.71**	**0.56**	**0.47**
	arima	0.88	0.82	0.83	0.78	0.87	0.81	0.78	0.79	0.86	0.81	0.78	0.79
	naive	1.20	1.22	1.19	1.21	1.20	1.22	1.19	1.21	1.20	1.22	1.19	1.21
regr01	seq2seq	0.91	0.89	0.88	**0.80**	0.92	0.90	0.87	0.80	0.89	0.88	0.87	**0.77**
	transf	**0.87**	0.87	**0.86**	**0.80**	0.89	0.87	**0.86**	0.79	**0.87**	0.87	0.86	0.78
	seq2one	0.89	0.88	0.89	0.87	0.90	0.88	0.87	0.83	**0.87**	0.86	0.86	0.78
	gbm	0.88	**0.86**	0.87	0.85	0.87	**0.85**	**0.86**	0.84	0.88	0.86	0.86	0.83
	mlp	0.91	0.91	0.92	0.90	0.90	0.91	0.91	0.87	0.90	0.89	0.89	0.84
	deepvar	0.87	**0.86**	**0.86**	0.86	**0.86**	0.88	0.87	0.85	0.88	**0.85**	**0.84**	0.84
	nhits	0.88	0.87	0.87	0.87	0.87	0.86	0.87	0.87	0.89	0.88	0.85	0.86
	mfbvar	0.91	0.92	0.90	0.84	0.90	0.91	0.90	**0.78**	0.89	0.90	0.88	0.79
	arima	0.90	0.88	0.88	0.89	0.87	0.87	0.86	0.87	0.88	0.86	0.85	0.86
	naive	1.24	1.25	1.23	1.23	1.24	1.25	1.23	1.23	1.24	1.25	1.23	1.23
regr02	seq2seq	0.88	0.88	0.86	**0.82**	0.88	0.88	0.86	**0.81**	0.87	0.87	0.85	0.81
	transf	0.89	0.87	0.86	0.83	0.90	0.88	0.86	0.82	0.88	0.88	0.86	**0.79**
	seq2one	**0.86**	**0.86**	**0.85**	0.86	0.86	**0.86**	**0.85**	0.86	**0.86**	0.86	**0.85**	0.86
	gbm	0.87	0.86	0.86	0.85	**0.85**	0.86	**0.85**	0.85	**0.86**	0.87	0.85	0.85
	mlp	0.94	0.95	0.95	0.95	0.90	0.91	0.91	0.90	0.89	0.90	0.89	0.88
	deepvar	**0.86**	**0.86**	**0.85**	0.86	0.86	0.87	0.87	0.85	0.87	0.87	0.85	0.86
	nhits	0.88	0.87	0.88	0.88	0.86	0.87	0.87	0.87	0.88	**0.85**	0.86	0.86
	mfbvar	0.94	0.95	0.94	0.87	0.92	0.93	0.89	**0.81**	0.89	0.90	0.87	0.80
	arima	0.90	0.89	0.90	0.88	0.88	0.87	0.89	0.88	0.87	0.86	0.86	0.86
	naive	1.21	1.22	1.20	1.19	1.21	1.22	1.20	1.19	1.21	1.22	1.20	1.19

2015).[8] Note that due to numerical instabilities encountered while running `mfbvar` on the aforementioned full set of data, we resort to using a significantly smaller dataset according to Schorfheide and Song (2021); see Table E.11. For all models, we conduct rolling forecasts/nowcasts up to one year ahead starting from 2018, with the models dynamically trained based on data up to the previous quarter-end. In the case of nowcasts, additional high-frequency data observations within the immediate next quarter of interest are included. Note that for the MIDAS regression model, we consider only the Almon polynomial constraining scheme (`almon`) and supply the true values of the HF block to produce predictions for the LF one, in a similar way to how it is handled in synthetic data experiments. The dataset contains a total of 249 quarterly and 747 monthly observations. Each dynamic model training relies on 200+ training samples. Such a small sample size becomes a bottleneck for calibrating NN-based models and may adversely impact their performance, as corroborated by the synthetic data experiments.

[8] The same implementation platforms as in Section 4.1 were used in the analysis.

Table 4
MAE of one and two-step-ahead forecasts/nowcasts for GDPC1 Q/Q AR, 1Q18 to 1Q22. The last column depicts the p-values of the Diebold–Mariano test, when comparing the forecasts of the seq2seq model vs its competitors, for the (rollling) one-step-ahead forecast vintage.

	One-step-ahead				Two-step-ahead				DM-test p-val
	F	N_1	N_2	N_3	F	N_1	N_2	N_3	
seq2seq	1.39	1.19	1.39	1.29	2.32	2.48	2.70	2.11	–
transf	1.57	1.59	1.73	1.77	1.69	1.64	1.93	1.60	0.27
seq2one	1.53	1.18	1.19	1.00	1.58	1.39	1.44	1.19	0.09
gbm	1.87	1.11	1.53	1.44	2.78	3.12	2.91	2.73	0.12
mlp	2.03	0.69	2.46	2.66	1.62	1.13	1.86	1.50	0.09
mfbvar	2.13	1.95	2.00	2.38	2.44	3.43	2.98	2.08	$<$1e−3
almon	3.84	4.64	2.05	4.01	3.62	2.87	2.53	4.11	$<$1e-4

The evaluation is based on the median absolute error (MAE) of the predictions against the true realized values for the period spanning from 1Q18 to 1Q22, which can be further divided into three sub-periods: (i) pre-Covid-19 (1Q18-4Q19), (ii) Covid-19 (1Q20-4Q20), and (iii) the Covid-19 recovery period (1Q21-1Q22). In particular, the Covid-19 pandemic disrupted the economy and led to significant shifts in production, investment, and consumption patterns (Advisers, 2022). We focus our discussion on the core economic indicators that characterize the state of the economy, namely GDPC1 (gross domestic product, quarter-over-quarter annualized rate or Q/Q AR, quarterly variable), CPILFESL (core consumer price index, year-over-year change or Y/Y, quarterly variable) and UNRATE (unemployment rate, levels, monthly variable). We also include the results for the Diebold–Mariano test (Diebold & Mariano, 2002), whose null hypothesis is that these forecast models/methods have comparable predictive accuracy. The p-values shown correspond to the (rolling) one-step-ahead forecast (vintage = F), when testing the predictive error of seq2seq against all other models considered. Further, Figure E.4 in Appendix E depicts boxplots of the p-values of the test for *all* low- and the high-frequency macroeconomic variables.

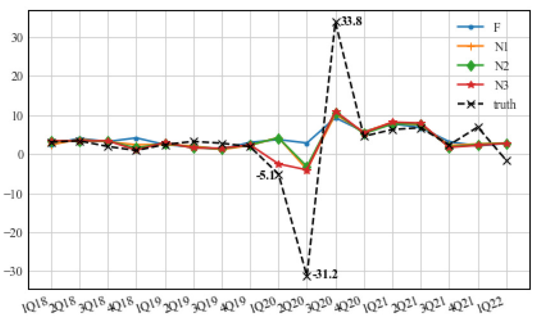
Table 4 displays the results for GDPC1, where the NN-based models (transf, seq2one, seq2seq) tend to outperform. In particular, transf and seq2one exhibit a strong performance across both prediction horizons, whereas that of seq2seq varies: it outperforms competing models for the one-step-ahead predictions, but falls behind for the two-step-ahead horizon. On the other hand, all ADL models and mfbvar exhibit much larger absolute deviations from actual measurements across both forecasting horizons. Regarding the DM test results, interestingly, the magnitude of the p-value largely aligns with the classes of models adopted. For example, it is as high as 0.27 between seq2seq and transformer, belonging to the proposed multi-task framework, whereas when the test is against linear models, the p-value becomes very small.

To gain further insight into the above observations, we examined the performance of the models over the three sub-periods. Focusing on the one-step-ahead predictions, all models exhibit better performance during the pre-Covid-19 period, with absolute deviation ranging from 0.5%–2.8%. On the other hand, large deviations (over 10%) are observed during the Covid-19 period. See, e.g., Fig. 2, which depicts the one-step ahead forecasts for GDPC1 across both forecast and nowcast vintages for the seq2seq and gbm models. The latter finding is expected, as the unprecedented shock in economic activity incurred observations of a magnitude that was unseen in the data. For example, GDPC1 (Q/Q AR) experienced a 31.2% drop in 2Q20, followed by a 33.8% rebound in 3Q20. The performance of all models improves during the Covid-19 recovery period, although it is still inferior to the pre-Covid-19 period. Note that the models based on the proposed framework outperform competing ones across all three sub-periods.

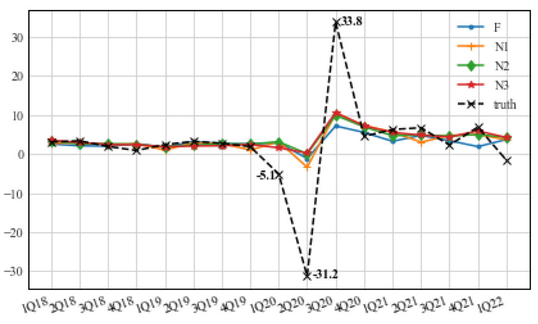
Table 5 presents the results for CPILFESL. It can be seen that gbm outperforms its competitors by a wide margin across both forecast and nowcast vintages (also evidenced by the corresponding small p-value from the Diebold–Mariano test), and almon exhibits strong performance in the forecasting vintage (F). This result is largely driven by its moderate advantage in performance during the Covid-19 period, while for the other two sub-periods, the proposed multi-task framework matches or slightly outperforms almon. Note that since CPILFESL is not one of the quarterly variables included for running mfbvar, a direct comparison is not applicable, so it is omitted.

Table 6 presents the results for UNRATE, for which mfbvar exhibits strong performance, followed by seq2seq. It is also worth noting that the performance of all models is better for the pre-Covid-19 and Covid-19 periods and deteriorates for the Covid-19 recovery period, which can be attributed to a shift in inflation patterns characterizing the latter period (Advisers, 2022). Finally, for the two-step-ahead predictions, seq2seq, seq2one, and almon exhibit similar performance overall.

Overall, seq2seq exhibits stable performance across the board, with a noticeable advantage over its competitors for GDPC1. On the other hand, transf's performance is uneven and varies across variables. Regarding competing models, seq2one exhibits good performance that follows closely that of seq2seq. Furthermore, mfbvar and almon perform well on certain variables (UNRATE and CPILFESL, respectively), whereas MLP can be characterized as the least competitive model across these three variables. Another general finding for all models is that the ingestion of fresh monthly information does not lead to significant improvements across nowcast vintages, in contrast to what was observed in the synthetic data experiments. This is primarily driven by the Covid-19 period



(a) method=`seq2seq`



(b) method=`gbm`

Fig. 2. Rolling one-step-ahead prediction for GDPC1 (Q/Q, AR), 1Q18 to 1Q22.

Table 5
MAE of one and two-step-ahead forecasts/nowcasts for Y/Y `CPILFESL`, 1Q18 to 1Q22. The last column corresponds to the DM-test *p*-value for `seq2seq` vs. others, for the (rolling) one-step-ahead forecast vintage.

	One-step-ahead				Two-step-ahead				DM-test *p*-val
	F	N_1	N_2	N_3	F	N_1	N_2	N_3	
seq2seq	0.56	0.59	0.64	0.61	0.84	0.87	0.89	0.94	–
transf	0.55	0.43	0.51	0.52	1.07	0.94	0.96	1.03	0.62
seq2one	0.54	0.40	0.53	0.46	0.81	0.77	0.76	0.65	0.38
gbm	0.24	0.22	0.23	0.20	0.26	0.25	0.28	0.27	0.001
mlp	0.69	0.69	0.72	0.57	1.40	1.13	1.00	0.91	0.22
almon	0.33	0.53	0.71	0.58	0.94	0.65	1.01	0.94	0.14

Table 6
MAE of the next quarter-end predictions of UNRATE, 1Q18 to 1Q22[a].

	F	N_1	N_2	N_3
seq2seq	0.49	0.43	0.49	–
transf	1.15	1.02	0.86	–
seq2one	0.65	0.66	0.69	–
gbm	0.59	0.42	0.30	–
mlp	1.38	1.34	1.36	–
mfbvar	0.27	0.29	0.23	–

[a]Note that given that UNRATE is a high-frequency variable and the setup has freq ratio 3, by construction, the next quarter-end values are observed for N_3.

and the recovery period, where rapid shifts in patterns in the data provide fluctuating signals to the models and thus may adversely impact nowcasts.

6. Discussion

The paper developed a multi-task encoder–decoder-based framework that efficiently addresses modeling tasks involving mixed-frequency data, leveraging recent developments in neural network architectures. In particular, it handles prediction tasks across different vintages in a unified manner, as new high-frequency data become progressively available during the period under consideration, without requiring model re-specification, retraining, or frequency alignment. Based on experiments on synthetic datasets with various different data generating mechanisms, and real datasets involving quarterly/monthly US macroeconomic indicators and electricity load and pricing data, the proposed framework demonstrated competitive performance vis-à-vis a number of competitors.

Note that the focus of the paper was to outline the major components—in their most basic form—involved in the proposed framework to efficiently handle mixed-frequency data prediction tasks. It is straightforward to enhance some of these components if needed in specific application domains, such as including learnable positional encoding (Liu, Yu, Dhillon, & Hsieh, 2020; Zhou et al., 2021), alternative encoder/decoder architectures (Wu, Xu, Wang, & Long, 2021), or adding additional modules, e.g., a temporal convolution network (or TCN, Bai, Kolter, & Koltun, 2018) to further process information in the encoded hidden states. Depending on the tasks in question, one could potentially adopt more advanced training strategies, e.g., having multiple optimizers so that the encoder/decoders' weights have different learning rates.

Finally, despite the well-documented advantages of transformer architectures in the literature, in both our synthetic and real data experiments, a transformer-based architecture underperformed when compared against an LSTM-based one. Note that a similar phenomenon has been observed in certain experiments pertaining to automatic speech recognition (Zeyer, Bahar, Irie, Schlüter, & Ney, 2019). The following two reasons are pertinent. First, a transformer does not posit any assumption on the (sequential) structure in the data. Any inter-dependencies are learned solely through the self-attention mechanism, which usually requires a large number of training samples. On the other hand, a sequential structure is embedded in an LSTM-based architecture. Consequently, it is conceptually more data-efficient for time series, due to the compatibility of its structure with respect to the properties (i.e., temporal dependence) of the data, analogously to how convolutional neural networks leverage the local structures embedded in images. Second, one advantage that is usually discussed in the literature (e.g., Vaswani et al., 2017) is that a transformer is better suited for handling long-range dependence. This is particularly relevant in natural language processing settings where contextual information often involves information in the remote past of the input. On the other hand, in settings with short-to-moderate dependence, such an advantage is harder to manifest. Recall that the data generating mechanisms used in the synthetic experiments did not exhibit long-range dependence, and this was also likely the case for both real datasets, where temporal dependence did not extend beyond a few lags.

7. Code and data availability

The implementation of the proposed methodology is available at the following repository: https://github.com/GeorgeMichailidis/multi-task-mixed-freqmulti-task-mixed-freq. Further, note that all real datasets used in the paper are publicly available. Instructions/scripts regarding how to obtain/process the data have been provided in the README in the above repository.

Declaration of competing interest

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

Acknowledgments

The authors would like to thank the editors and two anonymous referees for their constructive comments.

George Michailidis' work was supported in part by the US National Science Foundation under grants DMS-1821220 and DMS-2210358. He also acknowledges support from the University of Florida Research Computing HiPerGator system.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.ijforecast.2023.08.003.

References

Advisers, C. O. E. (2022). *Economic report of the president*. Washington, DC: US GPO, Annual.

Ankargren, S., Unosson, M., & Yang, Y. (2020). A flexible mixed-frequency vector autoregression with a steady-state prior. *Journal of Time Series Econometrics*, *12*(2).

Ankargren, S., & Yang, Y. (2021). Mixed-frequency Bayesian VAR models in R: the mfbvar package. *R Package Vignette*.

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

Bai, S., Kolter, J. Z., & Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv preprint arXiv:1803.01271.

Bao, W., Yue, J., & Rao, Y. (2017). A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PLoS One*, *12*(7), Article e0180944.

Baytas, I. M., Xiao, C., Zhang, X., Wang, F., Jain, A. K., & Zhou, J. (2017). Patient subtyping via time-aware LSTM networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 65–74).

Bell, V., Co, L. W., Stone, S., & Wallis, G. (2014). Nowcasting UK GDP growth. *Bank of England Quarterly Bulletin*, Q1.

Binkowski, M., Marti, G., & Donnat, P. (2018). Autoregressive convolutional neural networks for asynchronous time series. In *International conference on machine learning* (pp. 580–589). PMLR.

Chakraborty, N., Khare, K., & Michailidis, G. (2023). A Bayesian framework for sparse estimation in high-dimensional mixed frequency vector autoregressive models. *Statistica Sinica*, *33*, 1629–1652.

Challu, C., Olivares, K. G., Oreshkin, B. N., Garza, F., Mergenthaler, M., & Dubrawski, A. (2022). N-HiTS: Neural hierarchical interpolation for time series forecasting. In *The 37th AAAI conference on artificial intelligence*.

Che, Z., Purushotham, S., Cho, K., Sontag, D., & Liu, Y. (2018). Recurrent neural networks for multivariate time series with missing values. *Scientific Reports*, *8*(1), 1–12.

Cho, K., van Merrienboer, B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., et al. (2014). Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the empiricial methods in natural language processing*. (EMNLP).

Cimadomo, J., Giannone, D., Lenza, M., Monti, F., & Sokol, A. (2022). Nowcasting with large Bayesian vector autoregressions. *Journal of Econometrics*, *231*, 500–519.

Cross, J. L., Hou, C., & Poon, A. (2020). Macroeconomic forecasting with large Bayesian VARs: Global-local priors and the illusion of sparsity. *International Journal of Forecasting*, *36*(3), 899–915.

Dabrowski, J. J., & Rahman, A. (2019). Sequence-to-sequence imputation of missing sensor data. In *Australasian joint conference on artificial intelligence* (pp. 265–276). Springer.

Diebold, F. X., & Mariano, R. S. (2002). Comparing predictive accuracy. *Journal of Business & Economic Statistics*, *20*(1), 134–144.

Foroni, C., & Marcellino, M. G. (2013). A survey of econometric methods for mixed-frequency data. Available at SSRN 2268912.

Foroni, C., & Marcellino, M. (2014). A comparison of mixed frequency approaches for nowcasting euro area macroeconomic aggregates. *International Journal of Forecasting*, *30*(3), 554–568.

Gefang, D., Koop, G., & Poon, A. (2020). Computationally efficient inference in large Bayesian mixed frequency VARs. *Economics Letters*, *191*, Article 109120.

Ghysels, E. (2016). Macroeconomics and the reality of mixed frequency data. *Journal of Econometrics*, *193*(2), 294–314.

Ghysels, E., Kvedaras, V., & Zemlys, V. (2016). Mixed frequency data sampling regression models: the R package midasr. *Journal of Statistical Software*, *72*(1), 1–35.

Ghysels, E., Sinko, A., & Valkanov, R. (2007). MIDAS regressions: Further results and new directions. *Econometric Reviews*, *26*(1), 53–90.

Giannone, D., Reichlin, L., & Small, D. (2008). Nowcasting: The real-time informational content of macroeconomic data. *Journal of Monetary Economics*, *55*(4), 665–676.

Herzen, J., Lassig, F., Piazzetta, S. G., Neuer, T., Tafti, L., Raille, G., et al. (2022). Darts: User-friendly modern machine learning for time series. *Journal of Machine Learning Research*, *23*(124), 1–6.

Hochreiter, S., Bengio, Y., Frasconi, P., Schmidhuber, J., et al. (2001). Gradient flow in recurrent nets: the difficulty of learning long-term dependencies.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

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

Hyndman, R. J., Koehler, A. B., Snyder, R. D., & Grose, S. (2002). A state space framework for automatic forecasting using exponential smoothing methods. *International Journal of Forecasting*, *18*(3), 439–454.

Jennings, E., Jones, S., Arvola, L., Staehr, P. A., Gaiser, E., Jones, I. D., et al. (2012). Effects of weather-related episodic events in lakes: An analysis based on high-frequency data. *Freshwater Biology*, *57*(3), 589–601.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., et al. (2017). LightGBM: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems*, *30*.

Kim, Y.-J., & Chi, M. (2018). Temporal belief memory: Imputing missing data during RNN training. In *Proceedings of the 27th international joint conference on artificial intelligence*. (IJCAI-2018).

Kuzin, V., Marcellino, M., & Schumacher, C. (2011). MIDAS vs. mixed-frequency VAR: Nowcasting GDP in the euro area. *International Journal of Forecasting*, *27*(2), 529–542.

Lamb, A. M., Alias Parth Goyal, A. G., Zhang, Y., Zhang, S., Courville, A. C., & Bengio, Y. (2016). Professor forcing: A new algorithm for training recurrent networks. *Advances in Neural Information Processing Systems*, *29*.

Lepot, M., Aubin, J.-B., & Clemens, F. H. (2017). Interpolation in time series: An introductive overview of existing methods, their performance criteria and uncertainty assessment. *Water*, *9*(10), 796.

Li, S. C.-X., & Marlin, B. (2020). Learning from irregularly-sampled time series: A missing data perspective. In *International conference on machine learning* (pp. 5937–5946). PMLR.

Lipton, Z. C., Kale, D., & Wetzel, R. (2016). Directly modeling missing data in sequences with RNNs: Improved classification of clinical time series. In *Machine learning for healthcare conference* (pp. 253–270). PMLR.

Liu, X., Yu, H.-F., Dhillon, I., & Hsieh, C.-J. (2020). Learning to encode position for transformer with continuous dynamical model. In *International conference on machine learning* (pp. 6327–6335). PMLR.

Luo, Y., Cai, X., Zhang, Y., Xu, J., et al. (2018). Multivariate time series imputation with generative adversarial networks. *Advances in Neural Information Processing Systems*, *31*.

Ma, K., & Leung, H. (2019). A novel LSTM approach for asynchronous multivariate time series prediction. In *2019 International joint conference on neural networks* (IJCNN), (pp. 1–7). IEEE.

Makridakis, S., & Hibon, M. (2000). The M3-competition: results, conclusions and implications. *International Journal of Forecasting*, *16*(4), 451–476.

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2020). The M4 competition: 100, 000 time series and 61 forecasting methods. *International Journal of Forecasting*, *36*(1), 54–74.

Makridakis, S., Wheelwright, S. C., & Hyndman, R. J. (2008). *Forecasting methods and applications*. John Wiley & Sons.

Mariano, R. S., & Murasawa, Y. (2003). A new coincident index of business cycles based on monthly and quarterly series. *Journal of Applied Econometrics*, *18*(4), 427–443.

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

McCracken, M., & Ng, S. (2020). *Fred-QD: A quarterly database for macroeconomic research*: *Tech. rep.*, National Bureau of Economic Research.

McCracken, M. W., Owyang, M., & Sekhposyan, T. (2015). *Real-time forecasting with a large mixed frequency Bayesian VAR*: *FRB St. Louis Working Paper (2015-30)*.

Michel, P., Levy, O., & Neubig, G. (2019). Are sixteen heads really better than one? *Advances in Neural Information Processing Systems*, *32*.

Salinas, D., Flunkert, V., Gasthaus, J., & Januschowski, T. (2020). DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, *36*(3), 1181–1191.

Schirmer, M., Eltayeb, M., Lessmann, S., & Rudolph, M. (2021). Modeling irregular time series with continuous recurrent units. arXiv preprint arXiv:2111.11344.

Schorfheide, F., & Song, D. (2015). Real-time forecasting with a mixed-frequency VAR. *Journal of Business & Economic Statistics*, *33*(3), 366–380.

Schorfheide, F., & Song, D. (2021). *Real-time forecasting with a (standard) mixed-frequency VAR during a pandemic*: *Tech. rep.*, National Bureau of Economic Research.

Shabani, A., Abdi, A., Meng, L., & Sylvain, T. (2022). Scaleformer: iterative multi-scale refining transformers for time series forecasting. arXiv preprint arXiv:2206.04038.

Shukla, S. N., & Marlin, B. M. (2021). Multi-time attention networks for irregularly sampled time series. arXiv preprint arXiv:2101.10318.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems* (pp. 3104–3112).

Uematsu, Y., & Tanaka, S. (2019). High-dimensional macroeconomic forecasting and variable selection via penalized regression. *The Econometrics Journal*, *22*(1), 34–56.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., et al. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, *30*.

Weerakody, P. B., Wong, K. W., Wang, G., & Ela, W. (2021). A review of irregular time series data handling with gated recurrent neural networks. *Neurocomputing*, *441*, 161–178.

Wu, H., Xu, J., Wang, J., & Long, M. (2021). Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems*, *34*, 22419–22430.

Xu, Q., Liu, S., Jiang, C., & Zhuo, X. (2021). Qrnn-midas: A novel quantile regression neural network for mixed sampling frequency data. *Neurocomputing*, *457*, 84–105.

Xu, Q., Zhuo, X., Jiang, C., & Liu, Y. (2019). An artificial neural network for mixed frequency data. *Expert Systems with Applications*, *118*, 127–139.

Yang, W., Tian, Z., & Hao, Y. (2022). A novel ensemble model based on artificial intelligence and mixed-frequency techniques for wind speed forecasting. *Energy Conversion and Management*, *252*, Article 115086.

Zeyer, A., Bahar, P., Irie, K., Schlüter, R., & Ney, H. (2019). A comparison of transformer and LSTM encoder decoder models for ASR. In *2019 IEEE automatic speech recognition and understanding workshop* (ASRU), (pp. 8–15). IEEE.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., et al. (2021). Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence, Vol. 35* (pp. 11106–11115).