

Multi-Horizon Echo State Network Prediction of Intraday Stock Returns

Giovanni Ballarin¹, Jacopo Capra², and Petros Dellaportas³

September 4, 2026

Stock return prediction is a problem that has received much attention in the finance literature. In recent years, sophisticated machine learning procedures have been shown to perform significantly better than conventional prediction techniques. One downside of these approaches is that they are often numerically expensive to implement, for both training and inference, because of their high complexity. We propose a return prediction framework for intraday returns at multiple horizons based on Echo State Network (ESN) models, wherein a large subset of parameters is drawn at random and never trained. The coefficients that do require training can be fitted by explicitly solving a rolling-window, ridge-regularized least squares problem. Our approach enjoys the flexibility of machine learning methods, has an inherently efficient implementation, and shows strong forecasting performance.

Keywords: High-frequency data, reservoir computing, signal construction

JEL: G17, C45, C53

¹Universität Sankt Gallen, Mathematics and Statistics Division, Switzerland – giovanni.ballarin@unisg.ch. GB is grateful for the support of the Center for Doctoral Studies in Economics and the Chair of Empirical Economics at the University of Mannheim, where part of this work was developed, as well as the Great Minds Postdoctoral Fellowship program at Universität Sankt Gallen. Many thanks go to Lyudmila Grigoryeva and Erik-Jan Senn for their helpful comments and suggestions.

²UCL, Department of Statistical Science, United Kingdom – jacopo.t.capra@gmail.com

³UCL, Department of Statistical Science, United Kingdom and Athens University of Economics and Business, Department of Statistics, Greece – p.dellaportas@ucl.ac.uk

1 Introduction

The prediction of time series is a topic of great relevance in many economic and financial settings. Researchers aim to devise more effective, reliable, and parsimonious forecasting methods and models. In the field of financial portfolio management, the prediction of stock features (such as price, return, volatility, and trading volume) is a key component of many trading strategies. Traditionally, linear models are used for this purpose, as they are straightforward to implement and analyze. Major developments in the fields of artificial intelligence, machine learning (ML), and statistical learning methods have, however, greatly expanded the universe of potential forecasting approaches. A variety of methodologies have been proposed to improve the prediction of stock market data (Kumbure et al., 2022). Efforts involving neural networks (NNs) date back to the 1990s; see e.g. Nicholas Refenes et al. (1994), Ghosn and Bengio (1996), Saad et al. (1998), and Zhang et al. (1998). In recent years, researchers have evaluated financial machine learning forecasting approaches from both empirical (Sirignano and Cont, 2019, Aleti et al., 2025) and theoretical perspectives. From the latter point of view, works such as Didisheim et al. (2023) and Kelly et al. (2024) have formally argued why and how more complex models can much improve prediction, the so-called “virtue of complexity” (Kelly et al., 2024). Another aspect to consider is that returns at different horizons may embed distinct predictive information, which an investor can exploit, e.g., by constructing portfolios jointly targeting asset returns at multiple horizons. Multi-period portfolio optimization is thus a potentially attractive extension of the seminal mean-variance optimization (MVO) model developed by Markowitz (1952). The classical MVO approach is based on a single-period model, and multi-period optimization can be seen as a generalization of the standard Markowitz setting; see Blake et al. (2003), Grinold (2007), Qian et al. (2007), Grinold (2010), Sneddon (2008), Gârleanu and Pedersen (2013).

In this work, we develop a novel, computationally efficient forecasting approach for intra-day, multi-horizon return prediction that is scalable to both (very) high-frequency data, as well as data-poor settings.¹ Given the effectiveness and flexibility of machine learning methods in capturing nonlinear relationships between data, and their success and recent widespread

¹In our context, a “*data-poor*” setting is one where, although potentially large volumes of data are available, only a small fraction of it can actually be used to fit a prediction model. This may be due, for example, to fast regime changes, missing data, or numerically expensive preprocessing steps.

adoption in computer science, statistics, and econometrics, we wish to apply ML techniques to the intra-day multi-horizon return prediction problem. Yet, models such as deep neural networks are generally both expensive to train and require large datasets. Instead, we consider a special class of recurrent-type neural networks (RNNs) that are only partially trained: The recurrent state parameters are kept fixed at their random initialization values. We leverage the so-called Reservoir Computing approach and apply *Echo State Network* (ESN) models (Jaeger, 2001, Lukoševičius and Jaeger, 2009). The choice of applying the Echo State Networks to an intraday systematic trading context is also motivated by the empirical success ESNs have had in prediction tasks attempted in other fields (Tanaka et al., 2019). Our methodology builds on the recent advances in using ESNs for macroeconomic forecasting at different horizons and frequencies (Ballarin et al., 2024). When using a linear readout layer, ESNs can be fitted to data via regularized linear regression, which has a closed-form solution, unlike the computationally intensive stochastic backpropagation procedures used for neural networks. Three properties of ESN models are particularly useful in our setting: the *expressivity* of Echo State Network models, which are universal in terms of approximating nonlinear time series processes; ESN adaptability to *multi-horizon forecasting* by virtue of a nonlinear state formulation; the *computational efficiency* of ESNs, which lets one solve high-dimensional data problems with comparatively negligible computational overhead.

Our proposed ESN approach is predicated on first constructing a recurrent state equation that nonlinearly combines signals over time. The target returns are then predicted with a linear function of the ESN states, again using ridge-regularized least squares. For our regularized models, we employ a within-day rolling window cross-validation strategy to select hyperparameters. This setup allows us to systematically evaluate the performance gains of including regularization in the linear model, and additionally those obtained as a result of the ESN model.

In our application, we target 1290 U.S. stock price returns across 5 different intraday horizons at 10-minute intervals within the trading session. Our full forecasting pipeline can be run over one calendar year of data in a handful of minutes, highlighting the numerical efficiency of our ESN proposal. To measure forecasting effectiveness, we consider both mean squared forecasting error (MSFE) and the R^2 of predicted returns. As benchmarks, we employ two

linear forecasting models, as well as a machine learning model. The linear baselines are based on, respectively, ordinary linear regression over a vector of signals based on past market and stock performance, and a ridge-regularized linear model that shrinks coefficient estimates over the same set of signals. As ML benchmark, we also include a random forest model that takes the signals as inputs (Breiman, 2001). Our results show that our ESN-based forecasting strategy for the cross-section of returns dominates the benchmarks at all intraday horizons, with only the case of end-of-day (close) returns showing rather modest improvements. Notably, we achieve an up to 0.87% MSFE reduction that is also remarkably robust to the inherent model sampling randomness of ESN models. The random forest’s performance is, however, *worse* than that of the linear baseline by approximately the same absolute margin as the ESN. Per-horizon Diebold-Mariano and model confidence set (MCS) tests confirm that these gains are all statistically significant. Finally, to gauge the economic utility of our forecasts, we construct MV portfolios for a 10-minute horizon, non-overlapping returns, including a benchmark equal-weights portfolio. We find that ESN predictions yield the best gross cumulative returns and Sharpe ratio, and the lowest transaction costs and drawdowns.

Contributions. The core contributions of our work can be summarized as follows

- We provide a highly efficient, robust, and effective machine learning prediction framework for high-frequency returns that can be applied to very large asset panels. ESN model estimation and hyperparameter validation runtimes are comparable to those of (regularized) linear regression, enabling large-scale real-time model optimization.
- We rigorously test the performance of our proposed model against benchmarks, including random forests, which have been extensively used in the prediction of tabular data. Our empirical results show that ESNs achieve better forecasts than RFs at a much lower computation cost, and these prediction improvements also translate to higher returns and lower turnover in the respective MV strategies.
- Our empirical results provide additional insights on how predictability changes at different intraday horizons for a large cross-section of U.S. stocks. They suggest that this

phenomenon cannot be readily circumvented by relying on nonlinear prediction schemes, as their MSFEs eventually concentrate close to those of linear prediction models.

- Since, as we explain in detail in Section 4, ESN models have an inherently random nature, it is not known a priori how stable the forecasting performance is. While this aspect is often overlooked in practical applications of ESNs, we re-evaluate our empirical results for many different draws of the models. Our analysis shows that the prediction performance of our proposed approach is robust to model randomness.

Outline. In Section 2, we provide a review of the recent literature on ML methods in finance. Precise descriptions of the data, trading setting, and benchmark models are given in Section 3. Section 4 presents our proposed ESN model to forecast stock returns based on user-provided signals (we shall also use the word “features” interchangeably), as well as training procedures. Section 5, where we discuss our application to U.S. stock returns prediction over the year 2013, follows. Finally, Section 6 concludes.

2 Related Literature

Classical modeling of financial markets based on efficiency and risk neutrality suggests that stock returns should be unpredictable, and as such they are often econometrically modeled as random walks (Pesaran, 2015). Empirical finance research has nonetheless shown that this may not be entirely the case, with some evidence pointing towards the potential violation of these assumptions (Campbell et al., 2012). Focusing on intra-day returns specifically, Ait-Sahalia et al. (2025) recently provided extensive evidence of predictability at millisecond scales. Moreover, Aletti et al. (2025) showed that intraday returns may be effectively forecasted through the construction of a large collection of factors. We tackle a similar setting, with data sampled at 10-minute intervals, and find that at this coarser scale, too, there appears to be non-negligible return predictability.² It should be underscored that, compared to the earlier works mentioned below, to our knowledge, ours is the first to (i) consider a large universe of stocks, larger than

²In terms of *volatility* forecasting, Dhaene and Wu (2020) show that intraday (5-minute resolution) and overnight returns can also be incorporated in multivariate GARCH and BEKK-type models to considerably sharpen prediction.

e.g. the S&P500 components, that forms an *unbalanced* panel with missing observations, (ii) work at high (intra-day) frequencies, (iii) target multiple overlapping return horizons and (iv) implement a rolling-window cross-validation and fitting algorithm to allow for fast data regime changes.

ML Methods in Finance. There are many works in the past two decades that implement various machine learning techniques in a financial setting. [Olson and Mossman \(2003\)](#) and [Kwon and Moon \(2007\)](#) both employ modern neural network approaches to stock forecasting. [Yu and Yan \(2020\)](#) more recently employed Deep Neural Networks (DNNs) for the same task, finding that the flexibility afforded by deep models improves performance. [Borovkova and Tsiamas \(2019\)](#) use a Long Short-Term Memory (LSTM) recurrent architecture to classify stocks in high-frequency. [Tölö \(2020\)](#) exploits RNN-type networks to predict systemic crises in the financial sector. With a view towards the structural changes due to the COVID-19 pandemic, [Chandra and He \(2021\)](#) resort to a Bayesian NN approach to forecast prices of stocks from exchanges in the United States, Germany, China, and Australia, while [Abedin et al. \(2021\)](#) use a more standard deep learning approach to perform exchange rate prediction. Further, [Wang and Chen \(2024\)](#) propose a factor-based Generative Adversarial Network (GAN) to tackle both prediction and portfolio investment problems. For MVO specifically, [Kim et al. \(2024\)](#) also applied GANs: Their idea is to perform anomaly detection and augment the portfolio optimization problem with said information. [Vuletić et al. \(2024\)](#), using a novel “Fin-GAN” model, have shown that GANs trained with an economics-driven loss can be effective in achieving higher Sharpe ratios compared to both LSTM and ARIMA models. Lastly, it is important to mention that ML models, such as GANs and Variational Autoencoders, have also been extensively studied for synthetic financial data generation, see e.g. [Kim et al. \(2023\)](#), [Masi et al. \(2023\)](#), [Nagy et al. \(2023\)](#), [Acciaio et al. \(2024\)](#), [Kwon and Lee \(2024\)](#), and [Cetingoz and Lehalle \(2025\)](#) for a theoretical critique of this line of research.

Reservoir Computing Applications. Sophisticated neural network models are challenging to implement and train by construction. Echo State Networks are a recent development in the field of machine learning models that aim to tackle this issue, while still matching the

flexibility of fully-trainable neural networks. By avoiding training of state parameters, ESNs and RC models are able to circumvent the issues associated with backpropagation-through-time associated with classical RNNs (Hochreiter, 1998, Vlachas et al., 2020). Echo state networks have been applied to a wide variety of contexts (Sun et al., 2024). For example, ESNs have been successfully employed to predict water levels (Coulibaly, 2010), electricity loads (Deihimi and Showkati, 2012), and power generation by renewable sources (Hu et al., 2020). Variations involving deep architectures have also been devised (Kim and King, 2020). In the context of economic forecasting, Ballarin et al. (2024) constructed a class of ESN models that can incorporate data sampled at different frequencies, outperforming state-of-the-art models when forecasting U.S. GDP growth. The literature on financial applications is, to the best of our knowledge, minimal. Lin et al. (2009, 2011) were among the first to apply ESNs to stock price forecasting at daily frequency. Their model specification included a noise component in the state (which is known to negatively affect memory of past inputs, c.f. Guan et al., 2025), and used a fixed train-test split. Works such as Zhang et al. (2013) and Tian and Pei (2024) explored different avenues of extension of the ESN approach—phase space transformations and swarm-type hyperparameter search, respectively—but always in small-scale empirical setups.³ Recently, Sharma et al. (2025) showcased the efficacy of ESNs in predicting daily Bitcoin closing prices. Liu et al. (2018) provide an ESN application to financial data, but primarily in the context of hyperparameter optimization, while Trierweiler Ribeiro et al. (2021) use ESN models for the prediction of stock return volatilities. (Maciel et al., 2014) used Echo State Networks on currency exchange rate forecasting and trading. Lastly, alternative models also based on randomization have also been considered; see e.g. Akyildirim et al. (2023) for *random signatures* applied to portfolios and Gonon (2023) for theoretical guarantees on *random shallow network* learning of Black-Scholes PDEs. To the best of our knowledge, we are the first to quantitatively analyze the application of ESNs to forecasting and portfolio construction problems for a large panel of stocks.

³Zhang et al. (2013) evaluate their approach only on a single stock (Microsoft Corporation), while Tian and Pei (2024) consider 10 U.S. and 10 Chinese stocks. Both papers use data collected at a daily frequency.

3 Data and Setup

Intraday stock data was sourced from AlgoSeek, a market-leading platform providing high-quality high-frequency trading data.⁴ The data provider provides access to up to 1-minute resolution OHLC (open, high, low, close price) bars built from the reported Trades of the SIP (consolidated) data feed, which includes all U.S. Exchanges and FINRA from January 2007 to October 2020. AlgoSeek bars cover the entire trading day:

- Pre-Market: 4:00:00 AM to 9:29:59 AM;
- Market: 09:30:00 AM to 4:00:00 PM;
- Extended hours: 4:00:01 PM to 8:00:00 PM.

Throughout our analysis, we focus only on prices observed during market hours and employ Eastern Standard Time (EST) data time-stamps, although actual time and date may be different for each exchange due to geographical location. Given the large volume of data associated with the 1-minute frequency, we use downsampled observations at a 10-minute resolution in our empirical exercises. Nonetheless, our forecasting approach can be readily scaled to higher-frequency data. For an extended description of the data and its preprocessing, we refer the reader to Appendix A.

3.1 Trading Environment

The design of our forecasting experiments aims to mimic the setup of an intraday trading book, which rebalances its position only on an intraday basis and does not keep open the positions overnight. Accordingly, we assume that a portfolio rebalance can take place every 10 minutes, from 9:30 AM to 3:50 PM. Until 3:50 PM, we run individual stock return predictions every 10 minutes. It is important to note that all our predictions are done over future *overlapping* return horizons. For example, at 9:30 AM, the 10-minute prediction horizon goes from the 9:31 AM price bar to the 9:41 AM price bar. On the other hand, the 30-minute prediction horizon goes from the 9:31 AM price bar to the 10:01 AM price bar.

⁴<https://www.algoseek.com/>

We target the following prediction horizons:

- *10 minutes* (10 min): 39 predictions per day over this horizon.⁵
- *30 minutes* (30 min): 37 predictions per day over this horizon.
- *60 minutes* (60 min): 34 predictions per day.
- *2 hours* (2 hours): 28 predictions per day.
- *End-of-day* (EOD): 39 predictions per day. Specifically, EOD returns are built from the price at a particular time-stamp and the daily closing price.

We do not carry out any return prediction at 4:00 PM, but rather assume that we fully exit our book positions, submitting our orders at 3:50 PM to be executed at the closing auction price. Accordingly, we do not target any continuous session close to auction and overnight return horizons.

3.2 Prediction Setup

The universe of tradeable stocks is indexed by $i = 1, \dots, N$. For each stock, we have access to both future returns, $r_{t+h}^{(i)}$, and a set of D signals, collected in a D -dimensional vector $Z_t^{(i)} = (Z_{1,t}^{(i)}, \dots, Z_{D,t}^{(i)})' \in \mathbb{R}^D$. We shall also assume that D is fixed over time. Note that $r_{t+h}^{(i)}$ is indexed by horizon h , which for our purposes is an arbitrary intraday interval of time. For example, at any given time t , we write $h = 30$ min to indicate the 30-minute-ahead return, and $h = \text{EOD}$ for the end-of-day return. If the i -th stock return or any of its signals are not available at time t , we discard it for period t in our empirical exercise.⁶ Therefore, we assume that the *observable* cross-section at time t is composed of stocks with indices in set $\mathcal{I}_t \subseteq \{1, \dots, N\}$ of size $N_t := |\mathcal{I}_t|$, $N_t \leq N$.

⁵Returns are constructed at 10-minute intervals from 9:30 AM to 3:50 PM, which is the last time-stamp available for an intraday 10-minute horizon prediction. At 3:50 PM, the targeted return takes the closing auction price into account for the return calculation. The number of possible return predictions for other horizons is similarly computed, apart from end-of-day returns. For the EOD horizon, returns are still constructed every 10 minutes, but they are overlapping and cannot be observed within the trading day.

⁶Numerically, if a stock drops from the panel at a certain date or data is missing, we use **NaNs** to represent its signals and/or returns for that time index.

Our per-horizon forecasting objective is the prediction of the conditional expected return

$$\mathbb{E}[r_{t+h}^{(i)} \mid \mathcal{F}_t^{(i)}], \quad \text{where} \quad \mathcal{F}_t^{(i)} := \sigma(Z_t^{(i)}, Z_{t-1}^{(i)}, \dots)$$

and $\sigma(Z_t^{(i)}, Z_{t-1}^{(i)}, \dots)$ is the sigma algebra generated by all past realizations of $Z_t^{(i)}$, that is, the history of stock-specific signals. We are going to assume throughout that $r_{t+h}^{(i)}$ is measurable with respect to this filtration, and that any observable predictive information associated with past returns is already fully embedded in the signals.

3.3 Signal Construction

Our primary goal in signal construction is to separate stock-specific features from the general market structure. Moreover, while we could, in principle, work with discrete signals, such as buy-to-open or sell-to-open, and combinations of these with continuous signals. For the sake of simplicity, we focus on the systematic construction of real-valued signals following [Avellaneda and Lee \(2010\)](#). We discuss in more detail the technical aspects of the construction of signals in [Appendix B](#).

Let $P_t^{(i)}$ be the individual asset price. First, assume that prices are determined as a continuous-time stochastic process. Returns can be decomposed into drift, market factor (systematic), and idiosyncratic components:

$$\frac{dP_t^{(i)}}{P_t^{(i)}} = a^{(i)}dt + \sum_{j=1}^J b_j^{(i)} F_{j,t} + dU_t^{(i)}, \quad (1)$$

where $a^{(i)}$ is the stock price drift, $\{F_{j,t}\}_{j=1}^J$ are the returns of J risk-factors associated to the market, $\{b_j^{(i)}\}_{j=1}^J$ are the J factor loadings and $dU_t^{(i)}$ is a stock-specific residual component. We then model the residual term as an Ornstein-Uhlenbeck (OU) process:

$$dU_t^{(i)} = \kappa^{(i)}(m^{(i)} - U_t^{(i)})dt + \sigma^{(i)}dW_t, \quad (2)$$

where W_t is the standard Wiener process. As in [Avellaneda and Lee \(2010\)](#), $\kappa^{(i)}$, $m^{(i)}$, and $\sigma^{(i)}$ are stock-specific parameters which are only slowly varying over time.

The return residual model given by (1)–(2) is implemented in discrete time using a two-step procedure. First, we extract the market factors from observed returns using principal

component analysis (PCA). For each stock i , we then estimate the factor regression

$$r_t^{(i)} = a^{(i)} + \sum_{j=1}^J b_j^{(i)} F_{j,t} + v_t^{(i)}, \quad (3)$$

and denote the corresponding estimated idiosyncratic return residuals by $\hat{v}_t^{(i)}$. For a chosen lookback window P , we construct the cumulative residual process

$$\hat{U}_{P,t}^{(i)} := \sum_{s=t-P+1}^t \hat{v}_s^{(i)}.$$

The continuous-time OU specification in (2) implies that, when observations are one sampling interval apart, its exact discrete-time representation is an AR(1) process. We therefore estimate

$$\hat{U}_{P,t+1}^{(i)} = c_{0,P}^{(i)} + c_{1,P}^{(i)} \hat{U}_{P,t}^{(i)} + \eta_{P,t+1}^{(i)}. \quad (4)$$

Provided that $0 < c_{1,P}^{(i)} < 1$, the corresponding estimates of the mean-reversion rate and long-run mean are

$$\hat{\kappa}_P^{(i)} = -\log c_{1,P}^{(i)} \quad \text{and} \quad \hat{m}_P^{(i)} = \frac{c_{0,P}^{(i)}}{1 - c_{1,P}^{(i)}}.$$

Let

$$\hat{q}_P^{(i)} := \widehat{\text{Var}}(\eta_{P,t}^{(i)}).$$

denote the estimated innovation variance in (4). The stationary standard deviation of the fitted discrete-time process is then

$$\hat{\sigma}_{U,P}^{(i)} = \sqrt{\frac{\hat{q}_P^{(i)}}{1 - (c_{1,P}^{(i)})^2}}.$$

Consequently, the standardized residual, or *z-score signal*, associated with lookback window P is

$$z_{P,t}^{(i)} := \frac{\hat{U}_{P,t}^{(i)} - \hat{m}_P^{(i)}}{\hat{\sigma}_{U,P}^{(i)}}. \quad (5)$$

This signal measures the deviation of the stock-specific cumulative residual from its estimated long-run mean in units of its stationary standard deviation.

The factor model in (3) also contains the stock-specific drift $a^{(i)}$. To incorporate this component, note that the conditional drift of the idiosyncratic component in the continuous-time

model is

$$a^{(i)} + \kappa^{(i)}(m^{(i)} - U_t^{(i)}) = -\kappa^{(i)}\sigma_U^{(i)} \left[\frac{U_t^{(i)} - m^{(i)}}{\sigma_U^{(i)}} - \frac{a^{(i)}}{\kappa^{(i)}\sigma_U^{(i)}} \right].$$

This motivates the *modified z-score signal*

$$\tilde{z}_{P,t}^{(i)} := z_{P,t}^{(i)} - \frac{\hat{a}^{(i)}}{\hat{\kappa}_P^{(i)}\hat{\sigma}_{U,P}^{(i)}}, \quad (6)$$

where $z_{P,t}^{(i)}$ is as in (5) and $\hat{a}^{(i)}$ is obtained from (3). Thus, $\tilde{z}_{P,t}^{(i)}$ combines the displacement of the idiosyncratic component from its long-run mean with the stock-specific drift.

In our application, we estimate a total of 15 market factors.⁷ We build a total of six signals (stacked in vector $Z_t^{(i)}$) that differ in the level of discretization P . A core signal is computed following (6) and setting $P = 10$. We additionally built five alternative signals, also following (6), which differ from the core signal by setting $P \in \{20, 30, 60, 100, 150\}$.

3.4 Return Prediction Problem

Once signals are constructed, we can set up the return prediction problem. Under the simplifying assumption that the most recent signal observation contains sufficient information for forecasting, we have the general prediction equation

$$r_{t+h}^{(i)} = g_{t,h}(Z_t^{(i)}) + \epsilon_{t+h}^{(i)}, \quad (7)$$

where $g_{t,h} : \mathbb{R}^D \rightarrow \mathbb{R}$ is time and horizon-varying, possibly nonlinear function and $\epsilon_{t+h}^{(i)}$ is an error term. This setup is extremely broad, and estimation of $g_{t,h}$ is generally infeasible without additional restrictions on its functional form.

As a prediction baseline, we can consider a direct *linear* forecasting model, where future returns are predicted as an affine function of signals,

$$r_{t+h}^{(i)} = \mu_{t,h} + \beta'_{t,h} Z_t^{(i)} + \epsilon_{t+h}^{(i)}, \quad (8)$$

where $\mu_{t,h}$ is the market's average return, $\beta_{t,h} \in \mathbb{R}^D$ is the vector of feature coefficients, which is, in general, horizon and time-dependent. We make the standard assumption that $\beta_{t,h}$ does not

⁷To handle missing price observations, we forward-fill return data, which is equivalent to the standard assumption that when prices are unobserved, they remain fixed to the last observation. In total, the factors we use account for more than 90% of the variability of observed returns.

vary along the cross-section, that is, the signals we construct correctly embed any stock-specific information. Importantly, thanks to the linear structure of (8), $\mu_{t,h}$ and $\beta_{t,h}$ can be readily estimated by least-squares, yielding $\hat{\mu}_{t,h}$ and $\hat{\beta}_{t,h}$, respectively. The linear return prediction is thus $\hat{r}_{t+h}^{(i)} = \hat{\mu}_{t,h} + \hat{\beta}_{t,h}' Z_t^{(i)}$.

More sophisticated nonparametric and machine learning approaches may also be applied to estimate (7). Among these, common choices in the time series econometrics literature are kernel and series-based estimators (Bosq, 1996, Fan and Yao, 2003, Li and Racine, 2007), as well as neural networks, random forests, and boosting methods (Franses and van Dijk, 2008, Masini et al., 2023). To also provide a machine-learning-based benchmark, in our empirical exercise in Section 5, we use random forests (RFs) to construct a nonlinear estimate of (7). Random forests were introduced by Breiman (2001), and have since become a popular machine learning approach; see Biau and Scornet (2016) for a general introduction, Scornet and Hooker (2026) for theoretical properties, and Fallahgoul (2025) for a recent discussion of applications in finance.

4 Echo State Networks

The idea that random parameters may work well in complex nonlinear models such as artificial neural networks, in fact, goes back a long way (Lowe and Broomhead, 1988, Schmidt et al., 1992). Different research fields have developed this concept: Huang et al. (2006) proposed the “extreme learning machine” form for shallow NNs with randomly sampled internal weights;⁸ Rahimi and Recht (2008a,b) developed the “random kitchen sinks” approach (i.e., collections of functions with random parameters) in the setting of reproducing kernel Hilbert spaces (RKHS). More recent work on pinpointing the reasons behind the success of deep neural networks has also looked into the “lottery ticket” hypothesis (Frankle and Carbin, 2018, Malach et al., 2020, Ma et al., 2021), which informally states that in large NNs there is a high chance that a small subnetwork performs almost as well as its much larger super-network—even at initialization,

⁸By *shallow* we intend here that the number of hidden layers of the network is small, usually only one. However, neural networks with even a single layer are effective functional approximators. Textbook discussions of such results can be found, e.g., in Hastie et al. (2009), and also in Anthony and Bartlett (2009) or Zhang (2023) for formal theory.

before coefficients are trained (Sreenivasan et al., 2022). Works such as Zhao et al. (2022) and Bolager et al. (2023) go further and design special sampling schemes that try to optimize the initial weight sampling strategy, so that negligible-to-zero training is needed. In tandem, *reservoir computing* (RC) models have been developed as a (broad) family of methods which *explicitly* embrace that idea that most model parameters do not need to be trained, but can instead be randomly drawn from some distributions chosen by the researcher, independently of the data (Lukoševičius, 2012).⁹ Reservoir models, more specifically, originate from the fields of dynamic system analysis and control theory. An overarching subfamily of interest is thus that of Random Weight Neural Networks (RWNN), which encompasses all NN-like model designs where at least some of the weights are drawn randomly and not trained.

As mentioned in the introduction, an important consideration about RC and RWNN methods is their *computational efficiency*. Artificial neural network training often demands significant, if not extreme, investments related to both physical resources (computer clusters, hardware, etc.) and energy; these costs have been steadily rising over the past years as models quickly become more complex (Lohn and Musser, 2022). Since RC and RWNN models do not require full parameter training, the costs to optimize such models can be orders of magnitude lower. This also means that they can be effectively deployed in contexts where systematic re-training (e.g., online learning) is required.

4.1 ESN Modeling Framework

Echo State Networks are a relatively recent development in the field of nonlinear time series modeling. They belong to both reservoir computing and random weight network families. They are architecturally analogous to recurrent neural networks (RNN), in that they explicitly model the time-dependence of data using a state equation.

Formally, suppose that time series Y_t and Z_t are observed, and we seek to predict Y_{t+1} using Z_t and its past realizations (history). In our setting, Z_t is a vector of stock-specific signals, while Y_{t+1} is a stock return realized at a future horizon (e.g., 10-minute ahead). Let $Z_t = (Z_{1,t}, \dots, Z_{D,t})' \in \mathbb{R}^D$, where $D \geq 1$ is the number of stock signals, and $Y_t \in \mathbb{R}$. An ESN

⁹We remark that, while the coefficients are sampled at random, this does not mean that the models' *architecture* is also necessarily random. Indeed, taking into account time dependencies is essential.

model maps features Z_t to a *recurrent state* X_t defined by the equation

$$X_t := \alpha X_{t-1} + (1 - \alpha)\varphi(AX_{t-1} + CZ_t + b), \quad (9)$$

where $X_t \in \mathbb{R}^K$, for $K \geq D$; $A \in \mathbb{R}^{K \times K}$, $C \in \mathbb{R}^{K \times D}$ and $b \in \mathbb{R}^K$ are the state equation coefficients; $\varphi : \mathbb{R} \mapsto \mathbb{R}$ is a nonlinear, element-wise activation function (e.g. hyperbolic tangent or ReLU), and $\alpha \in [0, 1]$ is the state leak hyperparameter. Note that in the special setting where $K = D$, $\varphi(x) = x$, $A = 0_D$ is the $D \times D$ zero matrix, $C = I_D$ is the identity matrix of size D , $b = 0$ is the zero vector, and $\alpha = 0$, the above state equation simplifies to $X_t \equiv Z_t$. Therefore, by introducing state X_t we are considering a strictly more general class than that of linear models. In order to perform regression, the state must be mapped to an output value. The output of the ESN is determined by a linear layer,

$$X_t \mapsto \mu_t + \theta_t' X_t, \quad (10)$$

where $\theta_t \in \mathbb{R}^K$ are the output coefficients and $\mu_t \in \mathbb{R}$ is the output intercept term. Notice here that, in our ESN construction, we explicitly allow for the output coefficients to depend on time index t , while state parameters (A , B , and α) are constant. Thanks to the linear specification of (10), the model can be fitted by regularized least squares by solving

$$\{\hat{\mu}_t, \hat{\theta}_t\} := \arg \min_{\mu, \theta} \sum_{t=T_1}^{T_2} (Y_{t+1} - \mu - \theta' X_t)^2 + \phi(\theta), \quad (11)$$

where $\phi(\theta) > 0$ is the regularization term—e.g. $\phi(\theta) = \lambda \|\theta\|_2^2$ for ridge regression with an isotropic penalty—and we have assumed that a data sample over periods $T_1 \leq t \leq T_2$, $T_1 < T_2$, is available.

Remark 4.1. One can allow for more general state and output equations, and RC models with, for example, polynomial state equations and nonlinear output equations have been studied in the literature, c.f. [Gonon et al. \(2023\)](#). One issue with changing (10) to be nonlinear is that training can become much more complex. For example, letting the output layer be $X_t \mapsto g_\theta(X_t)$ where $g_\theta : \mathbb{R}^K \rightarrow \mathbb{R}$ is a neural network with parameters θ means that the ESN model will need to be trained by costly numerical optimization routines, e.g. stochastic gradient descent and its variants; hyperparameter tuning also becomes much more elaborate. Instead, we solve (11)

explicitly with cross-validated, ridge-regularized least squares (see Section 4.2 below).

Remark 4.2. In recent years, quantitative results on the approximation properties of ESNs and, more generally, reservoir computing models have been developed. Grigoryeva and Ortega (2018) and Gonon and Ortega (2020) have shown that ESNs are universal approximators of time filters acting on, respectively, deterministic sequences and stochastic processes. In the stochastic setting more specifically, Gonon et al. (2023) established that, if $Y_{t+1} = H^*(\mathbf{Z}_t)$, where $\mathbf{Z}_t := (Z_t, Z_{t-1}, \dots)$ and with $H^* : (\mathbb{R}^D)^{\mathbb{Z}^-} \rightarrow \mathbb{R}$ satisfying appropriate regularity conditions, there exists an ESN such that the L^2 approximation error $\mathbb{E}[(Y_t - H(\mathbf{Z}_t))^2]^{1/2}$ is of order $\mathcal{O}(1/\sqrt{K})$. This order of convergence is notably independent of the input dimension, D . Moreover, in Gonon et al. (2020) the same authors also developed generalization bounds for the out-of-sample error of ESN models.

Remark 4.3. While ESNs can be readily related to classical RNNs, there is not yet a universal framework to formally compare the two approaches. This is in part due to the fact that quantitative approximation theory for RNN-type models is still developing; see Wang et al. (2023), Jiao et al. (2025). From an applied perspective, ESNs and RNNs seem to have domain-dependent strengths and weaknesses, cf. Vlachas et al. (2020). It should also be noted that training of vanilla recurrent networks via gradient descent methods is known to be unstable, leading to the development of more robust (but not equivalent) alternatives, such as GRUs and LSTMs. Indeed, plain RNNs suffer greatly from exploding gradients during training (Hochreiter, 1998). This latter concern in particular motivates our comparison with random forests rather than a fully-trainable recurrent network in Section 5.

We underline here that simply drawing A , C , and b from some prescribed distribution will generally not yield an effective ESN model. This is because specific properties, such as the so-called *echo state* and *fading memory* properties (Grigoryeva and Ortega, 2018), are essential in guaranteeing the well-posedness and effectiveness of reservoir models. Therefore, we introduce additional hyperparameters, following the discussion in e.g. Ballarin et al. (2024). One begins by drawing random matrices A^* , C^* , and b^* . These are then normalized according to a specific matrix (pseudo)norm, $\|\cdot\|_A$, $\|\cdot\|_C$ and $\|\cdot\|_b$, respectively, yielding $\bar{A} := A^* / \|A^*\|_A$,

$\bar{C} := C^* / \|C^*\|_C$ and $\bar{b} := b^* / \|b^*\|_b$. The state equation then becomes

$$X_t := \alpha X_{t-1} + (1 - \alpha) \varphi(\rho \bar{A} X_{t-1} + \gamma \bar{C} Z_t + \zeta \bar{b}), \quad (12)$$

where $\rho \in [0, 1]$, $\gamma > 0$, and $\zeta \geq 0$ are the spectral radius, input scaling, and bias scaling, respectively. The reason why ρ is taken to be within the interval $[0, 1]$ is that $\|\cdot\|_A$ is most often chosen to either be the 2-norm or the largest absolute eigenvalue norm. We use the latter, so that the spectral radius of $\bar{A}$ is unity.

Remark 4.4. The way of sampling A^* , C^* , and b^* is itself a question of high practical importance. Many works have addressed this topic from an applied or computational perspective; see, e.g., [Lukoševičius \(2012\)](#) and references within. From a theoretical perspective, [Gonon et al. \(2023\)](#) derived explicit distributions for the state parameter matrices that depend on the structure of the approximation problem. In this work, we favor a practical perspective and apply some well-studied probability distributions. The entries of A^* are drawn from a sparse Gaussian distribution, while the entries of C^* are drawn from a sparse uniform distribution, since some empirical success with these has been observed in [Ballarin et al. \(2024\)](#). Moreover, in all of our ESN models, b is set to be the zero vector, without any random draw. Accordingly, we do not discuss the tuning of ζ in our application.

4.2 Return Prediction with ESN

We now specialize the general ESN model from the previous section to the return prediction setting of interest. We provide a diagram of the flow of information in our ESN model in [Figure 1](#). Following [Section 3.4](#), we do not instantiate N stock-specific ESN state equations, but rather draw a single set of state parameters that are then shared across stocks. Formally, we let the ESN state for the i th asset, $X_t^{(i)}$, be given by

$$X_t^{(i)} = \alpha X_{t-1}^{(i)} + (1 - \alpha) \varphi(\rho \bar{A} X_{t-1}^{(i)} + \gamma \bar{C} Z_t^{(i)}),$$

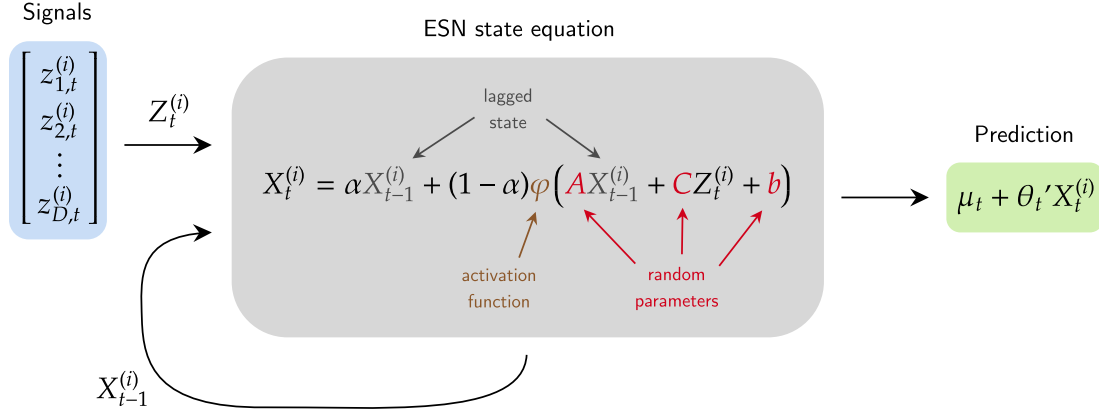


Figure 1: Overview of the proposed ESN model.

Notes: The signal vector is loaded into the ESN state equation and combined with the previous state via a nonlinear mapping with randomly sampled parameters. The prediction equation is linear in the ESN states, allowing for (regularized) least squares estimation. The state parameters do not change when fitting the model, and only μ_t , θ_t are estimated.

where φ , α , $\overline{A}$, $\overline{C}$, ρ and γ are fixed and identical across $i \in \{1, \dots, N\}$. The stock-specific ESN regression equation thus becomes

$$r_{t+h}^{(i)} = \mu_{t,h} + \theta_{t,h}' X_t^{(i)} + \varepsilon_{t+h}^{(i)}.$$

Observe that $\mu_{t,h}$ and $\theta_{t,h}$ are also shared across stocks. As with the linear model, here too we assume that returns can be predicted from signals with regression coefficients that are common across the stock cross-section. Since returns are, in general, overlapping and we employ a direct prediction strategy, the same state $X_t^{(i)}$ is used for all horizons. To handle missing data, we introduce what we term *state decay*, meaning that we replace missing signals with zeros, and continue iterating the state equation forward—although the associated state is not included in the estimated data; see Appendix C for details.

Remark 4.5. Our choice to implement state decay instead of opting for data in- and back-filling operations, such as linear interpolation between missing values, is primarily due to two key concerns. First, since we work with overlapping returns and online model estimation, we want to avoid information leakage from future observations to the past. Secondly, the missingness patterns are very heterogeneous across individual stocks, sometimes with large gaps. This makes it hard to construct valid and robust infill strategies (Bryzgalova et al., 2025), which

may even have negative effects on performance (Chen and McCoy, 2024).

To train the ESN, our objective is to sequentially minimize the regularized least squares empirical risk,

$$\mathcal{R}_{t,h}(\mu, \theta) := \frac{1}{M_{t,h}} \sum_{s=t-\tau_h-M_{t,h}}^{t-\tau_h-1} \frac{1}{N_s} \sum_{i \in \mathcal{I}_s} \left(r_{s+h}^{(i)} - \mu - \theta' X_s^{(i)} \right)^2 + \theta' \Lambda_{t,h} \theta, \quad (13)$$

where $1 \leq M_{t,h} \leq t - \tau_h$ is the *training window size* for the current horizon, $0 \leq \tau_h \leq t - 1$ is the *training window buffer*, and $\Lambda_{t,h}$ is a positive semi-definite diagonal penalization matrix. Notice that $\mathcal{R}_{t,h}(\mu, \theta)$ is not written explicitly as a function of ESN state coefficients, as these are *not* subject to training/fitting. The ridge-type regularization penalty is encoded in the time-varying, horizon-specific matrix $\Lambda_{t,h}$.

To operationalize (13), $M_{t,h}$, τ_h and $\Lambda_{t,h}$ need to be set. The window size controls how far back the model can go to extract information from past realized signals. A small window means that a short panel of observations is utilized to optimize $\mathcal{R}_{t,h}(\mu, \theta)$, while a longer panel may include irrelevant information with respect to current market conditions. Therefore, we tune $M_{t,h}$ for each prediction horizon h in our empirics. The window buffer τ_h is necessary because we want to ensure that in our training, we consider the time structure of multi-period returns. If $h = 10$ min, then at time t we may fit the model using states starting from up to ten minutes ago; instead, if we consider $h = 60$ min, then clearly we can only construct training tuples $\{X_s^{(i)}, r_{s+h}^{(i)}\}$ where $s \leq t - h$, meaning the most recent states we can use are from time $t - 2\text{hr}$. If we want to avoid constructing an empirical estimation scenario where there is information leakage across time, imposing $\tau_h \geq h$ is necessary. Note that this implies that for horizon EOD we are forced to use only observations up to the *previous trading day*, which are valid for EOD returns. These considerations need to be taken into account when considering the change in predictive performance of all methods, as presented in Section 5 below. Lastly, the optimal ridge penalty could change substantially across both time and horizon. We use cross-validation to select $\Lambda_{t,h}$ in our empirical analysis, and we provide a detailed discussion of our implementation in Appendix C.

Table 1: ESN Model Specifications

Parameter		Horizon				
		10 min	30 min	60 min	2 hours	EOD
K	State dim.	100	100	100	100	100
α	Leak rate	0.9	0.2	0	0	0
	A sparsity	0.15	0.15	0.15	0.65	0.35
ρ	Spectral radius	0.4	0.6	0.6	0.6	0
	C sparsity	0.95	0.55	0.75	0.85	0.25
γ	Input scaling	0.005	0.005	0.005	0.005	0.015

Note: All hyperparameters have been selected based on optimizing MSFE predictive performance on the Q4 2012 data sample.

5 Forecasting Multi-horizon Returns

We carry out the forecasting experiment using a period of time (i.e., 12 months of data) which, although short, is still performance-representative, especially because we are dealing with higher frequency data. Therefore, one calendar year still involves a large number of data points (16000) within the associated sample. The total number of stocks included in our dataset over this period is $N = 1290$. All forecasts are made in rolling-window fashion, so that at any given time index t , past observations are used to fit (and tune) the models, which then predict returns for all horizons $t + h$ with $h \in \{10 \text{ min}, 30 \text{ min}, 60 \text{ min}, 2 \text{ hours}, \text{EOD}\}$.

ESN Models. To adapt our ESN model to each prediction horizon, our model specifications are determined as follows. The state dimension is fixed to $K = 100$ for all horizons. We also set the random seed to a fixed value, so that we always draw the same A^* and C^* , but vary hyperparameters ρ , γ , and α , as well as the sparsity degree for the entries of both matrices. We collect our specifications in Table 1. The final hyperparameters are selected by using a pre-sample from Q4 2012, over which we minimize the mean squared forecasting error at each specific return horizon. In Appendix C, we provide more details on the numerical implementation of the hyperparameter optimization problem.

Linear Models. We employ the linear-in-signals and random forest models mentioned in Section 3.4 as benchmarks for prediction. In our experiments, we make a further modeling distinction between a “simple” linear model estimated via ordinary least squares and a *ridge-regularized* linear model for predicting returns. Indeed, the linear specification is distinct from the particular estimation strategy, meaning that ridge regularization is a separate refinement. By also using a rolling batch, it may be possible to better capture temporary changes in the return and signal structure, include them in the penalization procedure, and thus improving prediction performance compared to a more naïve least squares approach.

We let the basic linear model be defined as in (8), and estimated its coefficients using OLS on the most recently observed signals (i.e., fixing the window size to $M_{t,h} = 1$). The risk thus simplifies to

$$\mathcal{R}_{t,h}^{\text{linear}}(\mu, \beta) := \frac{1}{N_t} \sum_{i \in \mathcal{I}_t} \left(r_{t-\tau_h-1+h}^{(i)} - \mu - \beta' Z_{t-\tau_h-1}^{(i)} \right)^2.$$

The linear ridge regression model is estimated similarly to the ESN model by considering

$$\mathcal{R}_{t,h}^{\text{ridge}}(\mu, \beta) := \frac{1}{M_{t,h}} \sum_{s=t-\tau_h-M_{t,h}}^{t-\tau_h-1} \frac{1}{N_s} \sum_{i \in \mathcal{I}_s} \left(r_{s+h}^{(i)} - \mu - \beta' Z_s^{(i)} \right)^2 + \beta' \Lambda_{t,h} \beta,$$

where the window size $M_{t,h}$ is set identically to the respective values for ESN fitting. In both cases, the window buffer τ_h is also chosen to be the same as in the ESN estimation. The penalty matrix $\Lambda_{t,h}$ is also selected using the same cross-validation scheme employed for the ESN model. As discussed below, in our experiments, we find that baseline and benchmark linear models have very close performance. This suggests that we can primarily assign the improved performance to the nonlinear recurrent design of our ESN model, not the ridge regularization included in the estimation problem.

Random Forest. To further evaluate the impact of nonlinearities, as mentioned in Section 3.4, we also include a random forest (RF) regression model (Breiman, 2001). In our setting, RF regression aims to recover a nonlinear function $z \mapsto g_{t,h}(z)$ of the extracted signals defined as in equation (7).

Random forests are a stochastic ensemble of regression trees, which are conceptually similar to our chosen reservoir computing approach, since ESNs also exploit randomness. Formally,

assuming $Z_t^{(i)} \in \mathcal{Z} \subseteq \mathbb{R}^D$ for all i and t , and letting $\{B_j\}_{j=1}^J$ be a partition of $\mathcal{Z}$,¹⁰ a regression tree is the local constant model

$$\psi(Z_t^{(i)}) := \sum_{j=1}^J b_j \mathbb{I}\{Z_t^{(i)} \in B_j\},$$

where $\{b_j\}_{j=1}^J$ is the regression coefficient for partition element B_j . A partition can be obtained by dividing the domain $\mathcal{Z}$ into cardinal-aligned rectangular regions by using a binary tree construction, and this is the approach taken by regression trees. RF regression is run by constructing $L > 1$ bootstrap resamplings of the original data, building a regression tree for each bootstrap sample, ψ_ℓ where $\ell = 1, \dots, L$, and then averaging, yielding

$$\Psi(Z_t^{(i)}) = \frac{1}{L} \sum_{\ell=1}^L \psi_\ell(Z_t^{(i)}).$$

The random forest prediction risk is thus given by

$$\mathcal{R}_{t,h}^{\text{rf}}(\Psi) := \frac{1}{N_t} \sum_{i \in \mathcal{I}_t} \left(r_{t-\tau_h-1+h}^{(i)} - \Psi(Z_{t-\tau_h-1}^{(i)}) \right)^2,$$

where once more we use the same window buffer τ_h as for ESN and linear models. Random forests have been successfully applied to economic time series forecasting, although at present there is not yet a satisfactory theoretical analysis of their properties when applied to dependent data (Masini et al., 2023). More broadly, tree-based and boosting methods have been shown to achieve state-of-the-art performance in with tabular (e.g., panel) data; see Shwartz-Ziv and Armon (2022), Grinsztajn et al. (2022). We use the RF implementation of the `scikit-learn` Python package (Pedregosa et al., 2011); additional details on RF training are given in Appendix C.

Numerical Efficiency. One important concern when applying ML models in a high-frequency setting is how efficiently these can be estimated on broadly available hardware. We run all code on a system with a 64-core AMD EPYC 7763 CPU and 512GB of RAM; whenever possible, estimation routines are allowed to use all available cores for parallel computation. In Table 2 we report the time necessary for each model to be fitted and yield forecasts over the entire sample. It is easy to see that linear and ridge predictions are the most efficient models, fol-

¹⁰A collection of sets $\{B_j\}_{j=1}^J$ is a partition of $\mathcal{Z}$ if $\cup_j B_j = \mathcal{Z}$ and $B_j \cap B_{j'} = \emptyset$ for any $j \neq j'$.

Table 2: Model Runtimes

Linear	Ridge	RF	ESN
56 seconds	89 seconds	2561 seconds	139 seconds
1×	1.58×	45.73×	2.48×

Notes: For ridge regression and ESN models, the reported time includes ridge penalty cross-validation.

lowed by ESN, which is approximately 2.5 times slower than the baseline linear regression. The most expensive model by far is the random forest regression, which takes roughly 45 times longer to run than the least squares forecasting scheme. Frequent re-estimation of the random forest is significantly more expensive compared to the regularized estimation problem of the ESN. The runtime comparison of Table 2 suggests that more complex ML models—such as fully-trainable feedforward/deep neural networks, or transformer-based time series foundation models (Jin et al., 2026)—would also be expensive to evaluate. For recurrent-type models, like RNNs, LSTMs, GRUs, and state-space models (SSMs), one faces the additional complexity due to training complexity (cf. Remark 4.3), missing observations, and possibly short time series for a large subset of stocks. In comparison, a core advantage of the proposed RC approach is that the ESN is both efficient in terms of computation and straightforward to apply, even with a few observations.

5.1 Results

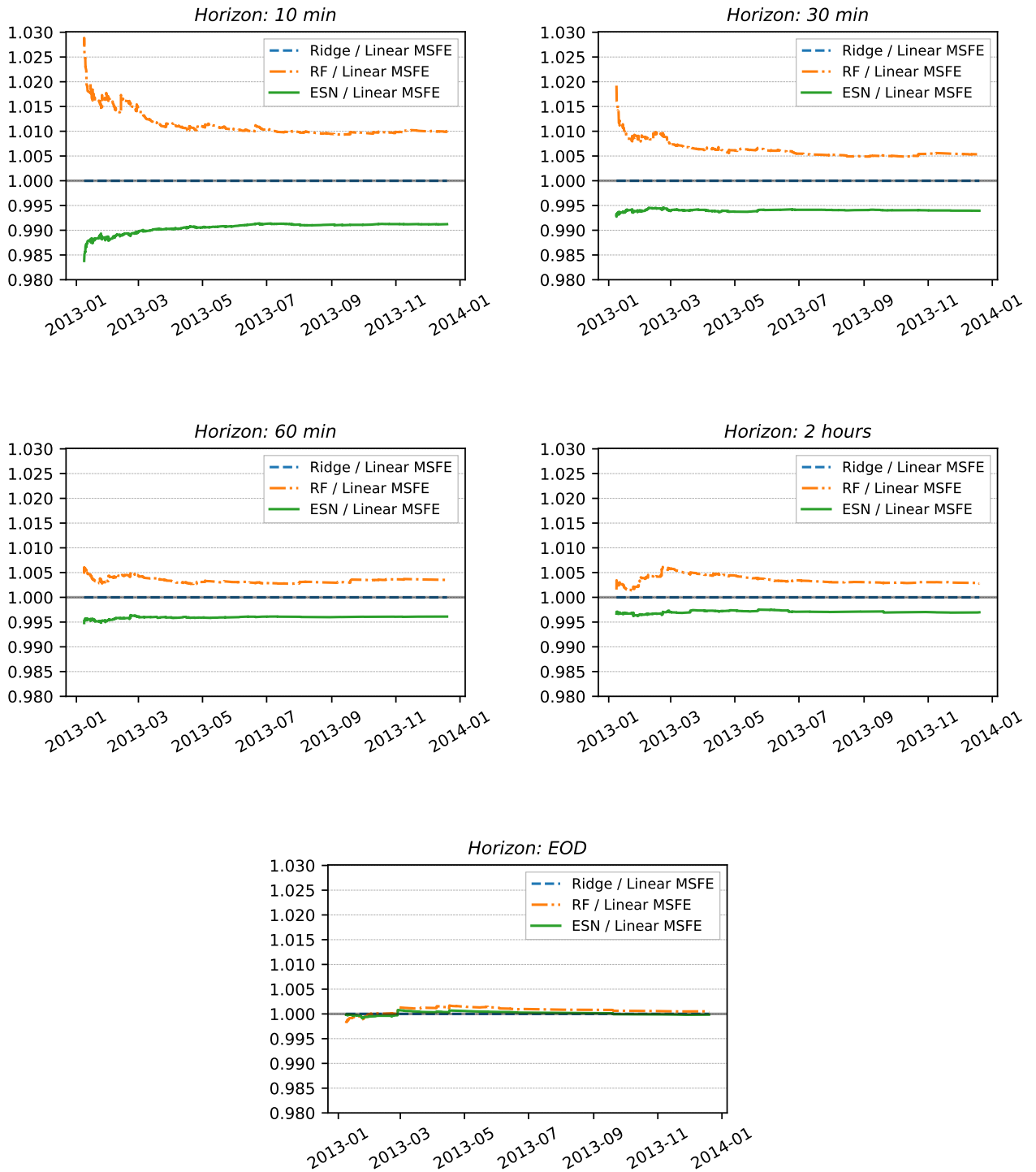
First, we consider the mean squared forecasting error (MSFE). At prediction horizon h and time t at which we carry out the prediction, the *instantaneous* MSFE across the N stocks in the investable universe at time t is given by

$$\text{MSFE}_{t,h} = \frac{1}{N_t} \sum_{i=1}^{N_t} (r_{t+h}^{(i)} - \hat{r}_{t+h}^{(i)})^2.$$

The lower the MSFE, the better the model’s performance in a mean-squared error sense. Over an investment period, we also consider the cumulative MSFE, $\text{cuMSFE}_{t,h} = \frac{1}{t} \sum_{s=1}^t \text{MSFE}_{s,h}$, as well as the total cumulated MSFE at terminal time T , that is $\text{MSFE}_h \equiv \text{cuMSFE}_{T,h}$.

Over the entire year of 2013, the cumulated MSFE of our ESN model is significantly lower

Figure 2: Relative Cumulative MSFE



Notes: Relative cumulative mean squared forecasting error with respect to linear model baseline. The first week of January 2013 is omitted to avoid initial fluctuations.

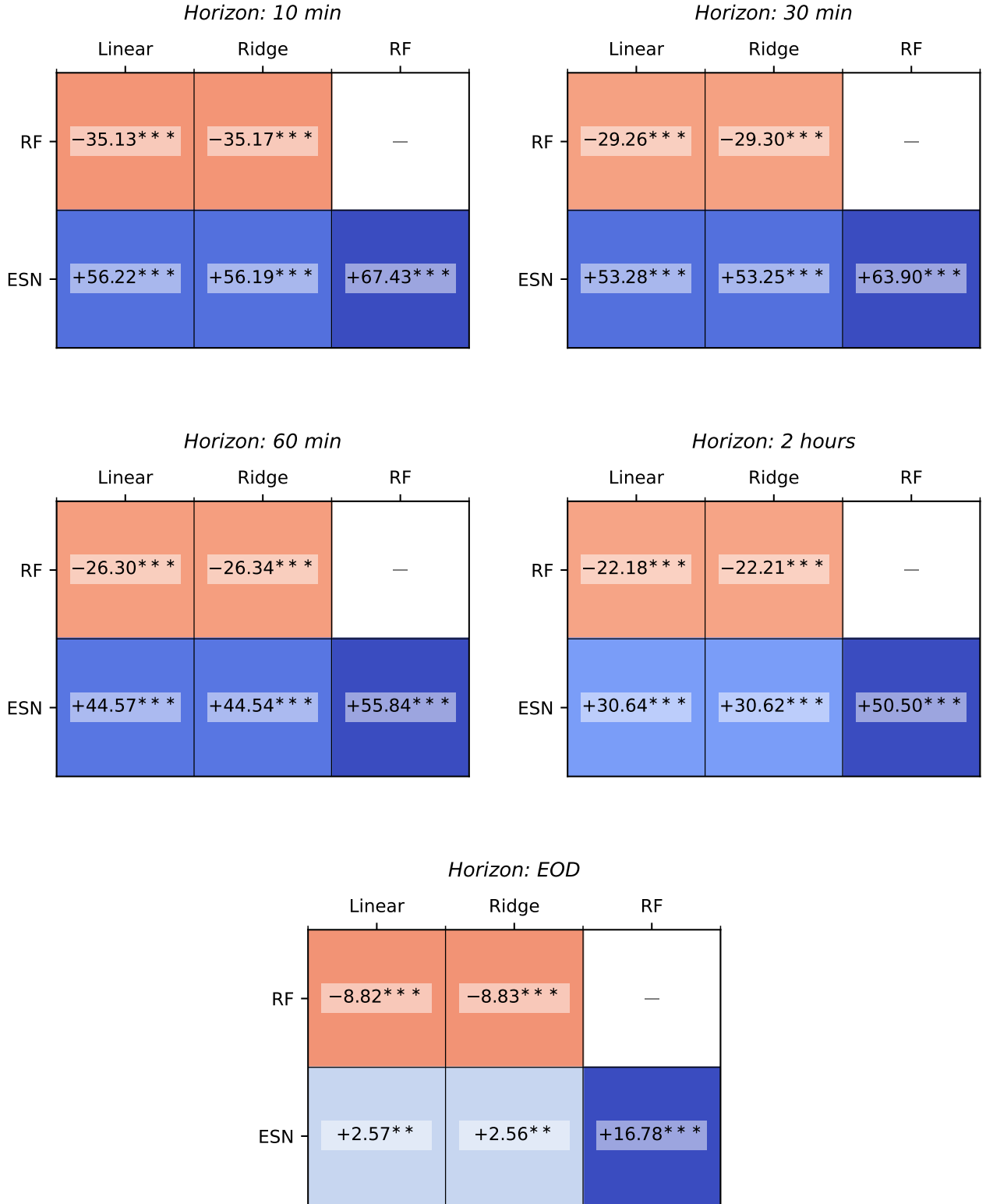
than either the baseline linear or ridge regression models, as can be seen from Table 3. Interestingly, the random forest model is measurably worse than both linear benchmarks, by approximately the same margin as the ESN model in absolute terms. In Figure 2, we also plot, for each of the five horizons we consider, the relative cumulative MSFE of all models—the baseline linear specification provides the reference point. In all cases, the difference between baseline and ridge is negligible: This is not exactly the case numerically, but MSFE deviations between the two models are insignificant for practical purposes. On the other hand, our ESN model is able to achieve a consistent reduction in MSFE of more than 0.87% at the 10-minute horizon. However, this gain, while still robust over time, diminishes progressively as the horizon grows. In fact, the gain essentially vanishes when we consider EOD returns, a setting where at times the cumulative error grows above that of the linear models. However, as evidenced in the last column of Table 3, even at EOD returns, the ESN can extract a relative improvement that is two orders of magnitude larger than that of simply applying regularization to the linear baseline.

In Table 4, we also present the total R^2 of forecasts with respect to return realizations. It is easy to see that, while all values are negative, ESN predictions show improvement, especially at 10, 30, and 60-minute horizons. We emphasize that, as Kelly et al. (2024) argue, negative R^2 's are not a sign of poor predictive or strategy efficacy. Accordingly, we find that, overall, our ESN framework attains better performance compared to our linear prediction benchmarks.

Performance Testing. Next, we argue that the gains enjoyed by our echo state network model are also significant from a statistical point of view. To test for the statistical significance of these improvements, we perform both pairwise Diebold-Mariano (DM) tests (Diebold and Mariano, 1995, Harvey et al., 1997) and Model Confidence Set (MCS) tests at all horizons (Hansen et al., 2011).

For the Diebold-Mariano tests, we focus only on comparing our ESN model with the benchmark and baseline linear predictions. Following the test results summarized in Figure 3, one can see that the null hypothesis of equal predictive ability is rejected extremely strongly—in particular, only at the end-of-day horizon, the p -values are above the 1% level. The MCS tests, on the other hand, enable comparison between all models simultaneously. As Table 5 shows, in

Figure 3: Diebold-Mariano Tests



Notes: Color matrix plots for one-sided Diebold-Mariano test statistics, column model versus row model. Statistical significance at 1% (***), 5% (**) and 10% (*) is shown next to statistic value.

Table 3: Total Cumulated MSFE

Model	Horizon				
	10 min	30 min	60 min	2 hours	EOD
Linear	0.0557	0.1402	0.2331	0.3704	0.7088
Ridge	0.0557 [-0.0010%]	0.1402 [-0.0007%]	0.2331 [-0.0004%]	0.3704 [-0.0003%]	0.7088 [-0.0001%]
RF	0.0562 [+1.0024%]	0.1409 [+0.5396%]	0.2340 [+0.3593%]	0.3714 [+0.2827%]	0.7092 [+0.0530%]
ESN	0.0552 [-0.8775%]	0.1393 [-0.6059%]	0.2322 [-0.3890%]	0.3693 [-0.3023%]	0.7087 [-0.0148%]

Notes: Percentage reduction in MSFE with respect to the linear model baseline is shown within squared brackets. Values are rounded to 4 decimal digits.

Table 4: Total Forecasting R^2 s

Model	Horizon				
	10 min	30 min	60 min	2 hours	EOD
Linear	-0.0766	-0.1213	-0.1128	-0.1656	-0.1760
Ridge	-0.0766	-0.1213	-0.1127	-0.1656	-0.1760
RF	-0.0870	-0.1227	-0.1167	-0.1689	-0.1766
ESN	-0.0675	-0.1146	-0.1084	-0.1621	-0.1758

Note: Values are rounded to 4 decimal digits.

all settings apart from EOD returns, our ESN models robustly dominate; with end-of-day data, however, both the ESN and the RF model are included in the best models at a 5% level. This outcome is not surprising, given the relative lack of improvement in cumulative MSFE shown in Figure 2 for the EOD horizon.

5.2 Robustness to Parameter Sampling

Since our Echo State Network prediction approach is inherently dependent on random draws of the state equation’s matrices, we check whether different draws of our model yield substantially different forecasting performance. To achieve this, we focus on 10-minute and 60-minute horizons and sample 100 distinct ESN models—meaning we redraw A^* and C^* with different

Table 5: Model Confidence Set Tests

Model	Horizon				
	10 min	30 min	60 min	2 hours	EOD
Linear	0	0	0	0	0.0011
Ridge	0	0	0	0	0.0002
RF	0	0	0	0	0.5766 [✓]
ESN	1 [✓]	1 [✓]	1 [✓]	1 [✓]	1 [✓]

Notes: Tests are run with the cross-sectional mean squared forecasting error using 10^4 bootstrap draws. The checkmark superscript ([✓]) is used to indicate inclusion of the method into the set of best prediction models at the 5% level.

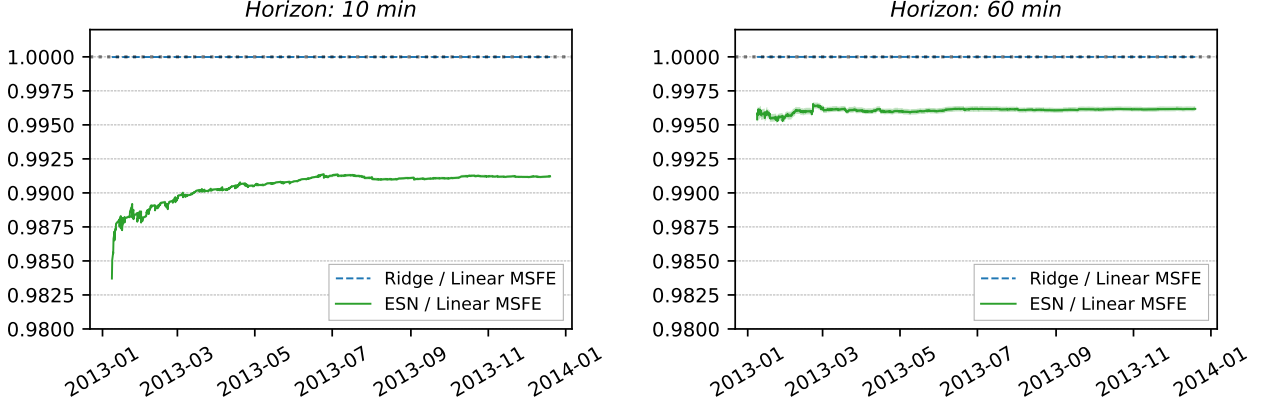
random seeds, while keeping unchanged the hyperparameters of Table 1—and re-run our empirical exercise. We provide relative cumulative MSFE plots showing 90% and 50% frequency bands, as well as median performance.

From Figure 4, we can see that the forecasting gains enjoyed by ESN models are remarkably robust to resampling state parameters. In fact, while we do draw shaded regions for relative cumulative MSFE interquantile ranges, these are very tight and thus hard to see, indicating that predictive performance is consistent. In both horizons, the MSFE metric is highly concentrated and robust to ESN model sampling randomness. Thus, we confirm that our approach yields robust gains over benchmark prediction models.

5.3 Portfolio Evaluation

To evaluate the economic implications of our forecasts, we use them in the construction of a classical mean-variance portfolio over the same testing period (Roncalli, 2016). We compare the ESN-based portfolio with benchmarks constructed using the linear, ridge, and random forests predictions, as well as a reference equal-weighted (EW) portfolio. For the sake of comparability, we subset the large panel of assets available to avoid including stocks that are thinly traded and, thus, may induce strictly mechanical changes in positions. This is particularly important when rebalancing at high frequencies. The final panel is has uniform cross section of size $\bar{N} = 388$. We focus on the 10-minute return horizon, as this setting does not require one to handle the

Figure 4: ESN Model Robustness



Notes: Light and dark shaded bands show 90% and 50% frequency regions, respectively, for ESN models. Median ESN performance is shown as a solid line. Note that for both horizons, the bands are thin and MSFE is concentrated around the median. The first week of January 2013 is omitted to avoid initial fluctuations.

time overlap that occurs at coarser horizons,¹¹ and shows the best improvement in MSFE for our ESN model; see Section 5.1. We provide a detailed discussion of asset selection and portfolio weight estimation in Appendix D.

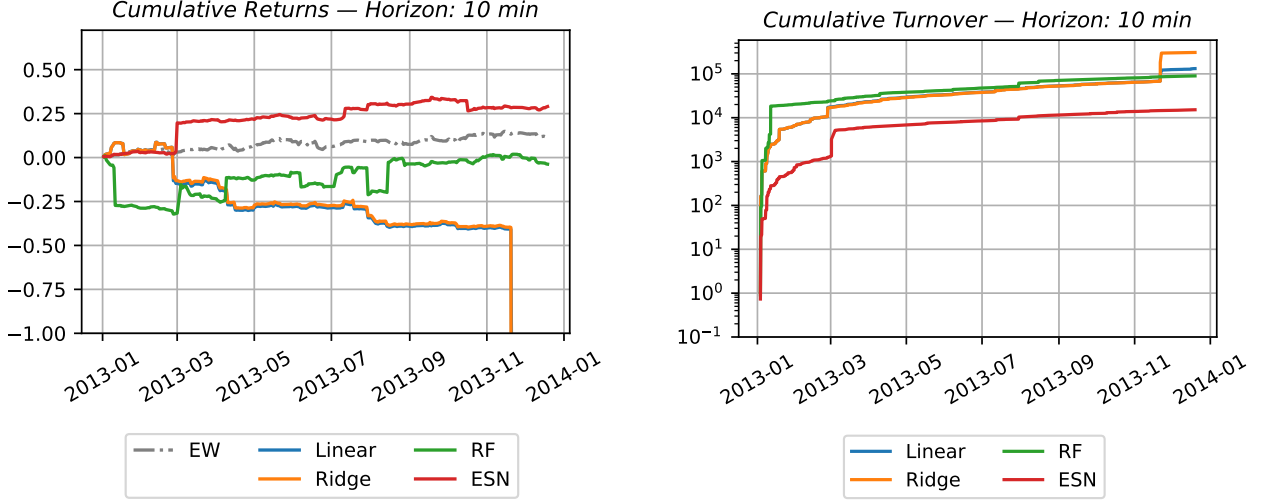
To determine the weights of portfolios based on return forecasts, we first require an estimate of the asset covariance matrix. As we only have 39 intraday realizations of 10-minute returns, the estimation problem is high-dimensional; moreover, since the data is high-frequency, we need to account for changes in correlation across days and weeks. Hence, we estimate the covariance matrix at a daily frequency using a nonlinear Ledoit-Wolf shrinkage estimator (Ledoit and Wolf, 2004). The previous day's covariance estimate, $\hat{\Sigma}_{[t]}$, is shared in the computation of the portfolio weights for all forecast methods, given by $\hat{\omega}_t^m = \hat{\Sigma}_{[t]}^{-1} \hat{r}_{t+10 \text{ min}}^m$, with $m \in \{\text{Linear, Ridge, RF, ESN}\}$. We normalize weights so that $\sum_{i=1}^N = 1$, and we initialize all portfolios with equal weights.

In Figure 5, we show cumulative returns for all portfolios, and the cumulated turnover for forecast-based strategies.¹² We can see that the ESN-based portfolio performs best, achieving raw returns higher than the benchmark EW strategy. Linear, ridge, and RF forecasts, on the

¹¹Intra-day overlapping returns at 10-minute frequency and hourly horizon, for example, require one to choose among 5 arbitrary starting times at the beginning of each day. Moreover, end-of-day returns do not allow for intra-day rebalancing, as EOD returns are only observed at market close.

¹²To plot cumulative returns, we use a reduced range for the y -axis due to instabilities in the linear and ridge portfolios. Figure D.2 in the appendix shows cumulative returns on an unrestricted axis.

Figure 5: Cumulative Portfolio Returns and Turnovers



Notes: Portfolio weights derived with linear, ridge, random forest, and ESN forecasts are computed on identical daily covariance matrix estimates with Ledoit-Wolf nonlinear shrinkage. Cumulative returns are net of transaction costs. Cumulative turnover is shown on a logarithmic axis.

other hand, produce volatile positions due to constant and substantial weight changes (see Figure D.1 in the Appendix). This is evidenced by much higher cumulative turnovers induced by these methods compared to the one achieved by the ESN portfolio weights. We also report Sharpe ratios and maximum drawdowns in Table 6. Among the forecast-based strategies, the ESN portfolio achieves the highest Sharpe ratio and the lowest maximum drawdown. Moreover, as Figure 5 shows, the ESN forecasts induce substantially lower portfolio turnover than the linear, ridge, and RF forecasts. An important distinction arises once transaction costs are taken into account, as the passive EW portfolio can outperform the actively rebalanced forecast-based strategies net of transaction costs (as shown in Figure D.3 in the Appendix). We do not view this result as contradicting the forecasting gains documented above. Rather, it illustrates the important distinction between statistical predictability and the economic implementability of a forecasting signal. At the 10-minute rebalancing frequency considered here, even relatively small changes in predicted returns can generate substantial changes in the weights of an unconstrained mean-variance portfolio, and consequently large transaction costs. In this respect, the substantially lower turnover generated by the ESN portfolio is itself noteworthy. Relative to the other forecast-based strategies, the ESN moves considerably towards converting improved predictive accuracy into a more stable portfolio allocation. Whether these forecasting

Table 6: Portfolio Sharpe Ratios and Maximum Drawdowns

	EW	Linear	Ridge	RF	ESN
Sharpe ratio	1.422	−1.181	−1.044	0.166	1.529
Max. drawdown	−0.743	−22.882	−85.350	−33.057	−0.465

Notes: Sharpe ratios are computed using total daily returns and annualized.

gains can be fully translated into superior net economic performance, however, depends also on the portfolio construction mechanism. A realistic implementation would typically incorporate transaction costs directly into the optimization problem or passive orders for some or all the assets (Ledoit and Wolf, 2025), and impose turnover, position, leverage, and other trading constraints. We deliberately do not introduce such constraints here, since our objective is to compare the economic implications of the different forecasting methods under the same simple portfolio mapping. Investigating the interaction between ESN forecasts and a fully constrained portfolio optimization framework is an important direction for future work.

6 Conclusion

In this work, we have presented a novel approach to forecast intraday stock returns at different horizons by leveraging ESN models. Our proposed method is nonparametric in nature and capable of significantly improving prediction performance across different horizons by a significant margin, with the only exception being end-of-day returns. Our findings suggest that there is potentially much to be gained by considering efficient, nonlinear forecasting models compared to regularized linear methods. Even though our ESN models inherently rely on randomly sampled parameters, we have shown that the prediction gains are robust to randomness at both short and medium intraday horizons. This robustness extends to comparison with alternative machine learning models, such as the widely-used random forest regressor. Additionally, we have established that ESN forecasting gains can translate to meaningful improvements in performance (compared, e.g., to equal-weighting schemes) when used to construct mean-variance portfolio strategies at a high-frequency, 10-minute horizon.

While it is beyond the scope of this work, one important thing we do not explore is whether our prediction method can also yield better performance—in terms of Sharpe ratio, for example—once applied specifically to *multi-horizon* MV portfolio construction. We consider this an important direction along which to extend our work, which nonetheless also requires the derivation of a realistic framework to jointly model overlapping returns and their forecasts. Market structure may also change frequently and suddenly, with potentially drastic changes in the volatility of and correlation between asset returns (Gottschalk, 2023). This is something that is indirectly observed in our portfolio analysis, and we expect that taking into account this factor may help reduce portfolio turnover. Moreover, as shown in our robustness analysis, it is straightforward to instantiate a large collection of diverse ESN models. Therefore, there is the possibility to combine our prediction framework with (online) model selection schemes, which can dynamically weight multiple ESNs to improve forecasting performance. For ESN models specifically, the ensemble approach was recently explored in Ballarin et al. (2025). Lastly, for the sake of simplicity and computational efficiency, we have constrained our ESN output layer to be linear. This is not a necessity, and it would be interesting to evaluate whether including more flexible output layers (e.g., a fully trainable shallow network) could further improve performance, and whether any gains obtained would be worthwhile given the increase in training complexity.

References

- Abedin, M. Z., Moon, M. H., Hassan, M. K., and Hajek, P. (2021). Deep learning-based exchange rate prediction during the COVID-19 pandemic. *Annals of Operations Research*, pages 1–52.
- Acciaio, B., Eckstein, S., and Hou, S. (2024). Time-Causal VAE: Robust Financial Time Series Generator. *Working Paper*.
- Ait-Sahalia, Y., Fan, J., Xue, L., and Zhu, X. (2025). How and When Are High-Frequency Stock Returns Predictable? *Management Science*, page mnscl.2022.02435.
- Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.
- Akyildirim, E., Gambarara, M., Teichmann, J., and Zhou, S. (2023). Randomized Signature Methods in Optimal Portfolio Selection. *Working Paper*.
- Aleti, S., Bollerslev, T., and Siggaard, M. (2025). Intraday Market Return Predictability Culled from the Factor Zoo. *Management Science*.
- Anthony, M. and Bartlett, P. L. (2009). *Neural Network Learning: Theoretical Foundations*. Cambridge University Press.
- Avellaneda, M. and Lee, J.-H. (2010). Statistical arbitrage in the US equities market. *Quantitative Finance*, 10(7):761–782.
- Ballarin, G., Dellaportas, P., Grigoryeva, L., Hirt, M., van Huellen, S., and Ortega, J.-P. (2024). Reservoir computing for macroeconomic forecasting with mixed-frequency data. *International Journal of Forecasting*, 40(3):1206–1237.
- Ballarin, G., Grigoryeva, L., and Li, Y. C. (2025). From Many Models, One: Macroeconomic Forecasting with Reservoir Ensembles. *Working Paper*, 10.48550/arXiv.2512.13642.
- Biau, G. and Scornet, E. (2016). A random forest guided tour. *TEST*, 25(2):197–227.

- Blake, C., Petrich, D., and Ulitsky, A. (2003). The Right Tool for the Job: Using Multiperiod Optimization in Transitions. *Trading*, 17(1):33–37.
- Bolager, E. L., Burak, I., Datar, C., Sun, Q., and Dietrich, F. (2023). Sampling weights of deep neural networks. *Advances in Neural Information Processing Systems*, 36:63075–63116.
- Borovkova, S. and Tsiamas, I. (2019). An ensemble of LSTM neural networks for high-frequency stock market classification. *Journal of Forecasting*, 38(6):600–619.
- Bosq, D. (1996). *Nonparametric Statistics for Stochastic Processes: Estimation and Prediction*. Number v.110 in Lecture Notes in Statistics Ser. Springer New York, New York, NY.
- Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1):5–32.
- Bryzgalova, S., Lerner, S., Lettau, M., and Pelger, M. (2025). Missing Financial Data. *The Review of Financial Studies*, 38(3):803–882.
- Campbell, J. Y., Lo, A. W., and MacKinlay, A. C. (2012). *The Econometrics of Financial Markets*. Princeton University Press, Princeton, NJ.
- Cetingoz, A. R. and Lehalle, C.-A. (2025). Synthetic Data for Portfolios: A Throw of the Dice Will Never Abolish Chance. *Working Paper*.
- Chandra, R. and He, Y. (2021). Bayesian neural networks for stock price forecasting before and during COVID-19 pandemic. *PLoS One*, 16(7):e0253217.
- Chen, A. Y. and McCoy, J. (2024). Missing values handling for machine learning portfolios. *Journal of Financial Economics*, 155:103815.
- Coulibaly, P. (2010). Reservoir Computing approach to Great Lakes water level forecasting. *Journal of Hydrology*, 381(1):76–88.
- Deihimi, A. and Showkati, H. (2012). Application of echo state networks in short-term electric load forecasting. *Energy*, 39(1):327–340.

- Dhaene, G. and Wu, J. (2020). Incorporating overnight and intraday returns into multivariate GARCH volatility models. *Journal of Econometrics*, 217(2):471–495. Nonlinear Financial Econometrics.
- Didisheim, A., Ke, S. B., Kelly, B. T., and Malamud, S. (2023). Complexity in Factor Pricing Models. *NBER Working Paper Series*.
- Diebold, F. X. and Mariano, R. S. (1995). Comparing predictive accuracy. *Journal of Business & Economic Statistics*, 13(3):253–263.
- Fallahgoul, H. (2025). High-Dimensional Learning in Finance. *Working Paper*, arXiv:2506.03780.
- Fan, J. and Yao, Q. (2003). *Nonlinear Time Series: Nonparametric and Parametric Methods*. Springer Series in Statistics. Springer New York, New York, NY.
- Frankle, J. and Carbin, M. (2018). The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. In *International Conference on Learning Representations*.
- Franses, P. H. and van Dijk, D. (2008). *Nonlinear Time Series Models in Empirical Finance*. Cambridge Univ. Press, Cambridge, 6. print edition.
- Gârleanu, N. and Pedersen, L. H. (2013). Dynamic Trading with Predictable Returns and Transaction Costs. *Journal of Finance*, 68(6):2309–2340.
- Ghosn, J. and Bengio, Y. (1996). Multi-Task Learning for Stock Selection. *Advances in Neural Information Processing Systems*, 9.
- Gonon, L. (2023). Random Feature Neural Networks Learn Black-Scholes Type PDEs Without Curse of Dimensionality. *Journal of Machine Learning Research*, 24(189):1–51.
- Gonon, L., Grigoryeva, L., and Ortega, J.-P. (2020). Risk Bounds for Reservoir Computing. *Journal of Machine Learning Research*, 21(240):1–61.
- Gonon, L., Grigoryeva, L., and Ortega, J.-P. (2023). Approximation bounds for random neural networks and reservoir systems. *The Annals of Applied Probability*, 33(1):28–69.

- Gonon, L. and Ortega, J. (2020). Reservoir Computing Universality With Stochastic Inputs. *IEEE Transactions on Neural Networks and Learning Systems*, 31(1):100–112.
- Gottschalk, S. (2023). From Black Wednesday to Brexit: Macroeconomic shocks and correlations of equity returns in France, Germany, Italy, Spain, and the United Kingdom. *International Journal of Finance & Economics*, 28(3):2843–2873.
- Grigoryeva, L. and Ortega, J.-P. (2018). Echo state networks are universal. *Neural Networks*, 108:495–508.
- Grinold, R. (2007). Dynamic Portfolio Analysis. *The Journal of Portfolio Management*, 34(1):12–26.
- Grinold, R. (2010). Signal Weighting. *The Journal of Portfolio Management*, 36(4):24–34.
- Grinsztajn, L., Oyallon, E., and Varoquaux, G. (2022). Why do tree-based models still outperform deep learning on typical tabular data? In *Advances in Neural Information Processing Systems*, volume 35, pages 507–520. Curran Associates, Inc.
- Guan, J., Kubota, T., Kuniyoshi, Y., and Nakajima, K. (2025). How noise affects memory in linear recurrent networks. *Physical Review Research*, 7(2):023049.
- Hansen, P. R., Lunde, A., and Nason, J. M. (2011). The Model Confidence Set. *Econometrica*, 79(2):453–497.
- Harvey, D., Leybourne, S., and Newbold, P. (1997). Testing the equality of prediction mean squared errors. *International Journal of Forecasting*, 13(2):281–291.
- Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, New York, NY, USA.
- Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(2):107–116.

- Hu, H., Wang, L., and Lv, S.-X. (2020). Forecasting energy consumption and wind power generation using deep echo state network. *Renewable Energy*, 154:598–613.
- Huang, G.-B., Chen, L., and Siew, C.-K. (2006). Universal approximation using incremental constructive feedforward networks with random hidden nodes. *IEEE transactions on neural networks*, 17(4):879–892.
- Jaeger, H. (2001). The “echo state” approach to analysing and training recurrent neural networks—with an erratum note. Technical Report 34, German National Research Center for Information Technology.
- Jiao, Y., Wang, Y., and Yan, B. (2025). Approximation Bounds for Recurrent Neural Networks with Application to Regression.
- Jin, M., Kong, Y., Liang, Y., Zhang, C., Xue, S., Wang, X., Zhang, J., Wang, Y., Chen, H., Li, X.-L., Tseng, V., Zheng, Y., Chen, L., Xiong, H., Pan, S., and Wen, Q. (2026). Large Models for Time Series and Spatio-Temporal Data: A Survey and Outlook. *ACM Computing Surveys*, 58(14):1–38.
- Kelly, B., Malamud, S., and Zhou, K. (2024). The Virtue of Complexity in Return Prediction. *The Journal of Finance*, 79(1):459–503.
- Kim, J. H., Kim, S., Lee, Y., Kim, W. C., and Fabozzi, F. J. (2024). Enhancing mean–variance portfolio optimization through GANs-based anomaly detection. *Annals of Operations Research*, pages 1–28.
- Kim, S., Hong, J., and Lee, Y. (2023). A GANs-Based Approach for Stock Price Anomaly Detection and Investment Risk Management. In *ACM Other conferences*, pages 1–9. Association for Computing Machinery, New York, NY, USA.
- Kim, T. and King, B. R. (2020). Time series prediction using deep echo state networks. *Neural Computing and Applications*, 32(23):17769–17787.

- Kumbure, M. M., Lohrmann, C., Luukka, P., and Porras, J. (2022). Machine learning techniques and data for stock market forecasting: A literature review. *Expert Syst. Appl.*, 197:116659.
- Kwon, S. and Lee, Y. (2024). Can GANs Learn the Stylized Facts of Financial Time Series? In *ACM Other conferences*, pages 126–133. Association for Computing Machinery, New York, NY, USA.
- Kwon, Y.-K. and Moon, B.-R. (2007). A Hybrid Neurogenetic Approach for Stock Forecasting. *IEEE Trans. Neural Networks*, 18(3):851–864.
- Ledoit, O. and Wolf, M. (2004). A well-conditioned estimator for large-dimensional covariance matrices. *Journal of Multivariate Analysis*, 88(2):365–411.
- Ledoit, O. and Wolf, M. (2025). Markowitz portfolios under transaction costs. *The Quarterly Review of Economics and Finance*, 100:101962.
- Li, Q. and Racine, J. S. (2007). *Nonparametric Econometrics: Theory and Practice*. Princeton University Press, Princeton, N.J.
- Lin, X., Yang, Z., and Song, Y. (2009). Short-term stock price prediction based on echo state networks. *Expert Systems with Applications*, 36(3, Part 2):7313–7317.
- Lin, X., Yang, Z., and Song, Y. (2011). Intelligent stock trading system based on improved technical analysis and Echo State Network. *Expert Systems with Applications*, 38(9):11347–11354.
- Liu, J., Sun, T., Luo, Y., Fu, Q., Cao, Y., Zhai, J., and Ding, X. (2018). Financial Data Forecasting Using Optimized Echo State Network. In Cheng, L., Leung, A. C. S., and Ozawa, S., editors, *Neural Information Processing*, Lecture Notes in Computer Science, pages 138–149, Cham. Springer International Publishing.
- Lohn, A. J. and Musser, M. (2022). AI and Compute: How Much Longer Can Computing Power Drive Artificial Intelligence Progress? Technical report, Center for Security and Emerging Technology.

- Lowe, D. and Broomhead, D. (1988). Multivariable functional interpolation and adaptive networks. *Complex systems*, 2(3):321–355.
- Lukoševičius, M. (2012). A Practical Guide to Applying Echo State Networks. In Montavon, G., Orr, G. B., and Müller, K.-R., editors, *Neural Networks: Tricks of the Trade: Second Edition*, Lecture Notes in Computer Science, pages 659–686. Springer, Berlin, Heidelberg.
- Lukoševičius, M. and Jaeger, H. (2009). Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149.
- Ma, X., Yuan, G., Shen, X., Chen, T., Chen, X., Chen, X., Liu, N., Qin, M., Liu, S., Wang, Z., and Wang, Y. (2021). Sanity Checks for Lottery Tickets: Does Your Winning Ticket Really Win the Jackpot? In *Advances in Neural Information Processing Systems*, volume 34, pages 12749–12760. Curran Associates, Inc.
- Maciel, L., Gomide, F., Santos, D., and Ballini, R. (2014). Exchange rate forecasting using echo state networks for trading strategies. In *2014 IEEE Conference on Computational Intelligence for Financial Engineering & Economics (CIFER)*, pages 40–47.
- Malach, E., Yehudai, G., Shalev-Schwartz, S., and Shamir, O. (2020). Proving the Lottery Ticket Hypothesis: Pruning is All You Need. In *Proceedings of the 37th International Conference on Machine Learning*, pages 6682–6691. PMLR.
- Markowitz, H. (1952). Portfolio Selection. *Journal of Finance*, 7(1):77–91.
- Masi, G., Prata, M., Conti, M., Bartolini, N., and Vyetrenko, S. (2023). On Correlated Stock Market Time Series Generation. In *ACM Other conferences*, pages 524–532. Association for Computing Machinery, New York, NY, USA.
- Masini, R. P., Medeiros, M. C., and Mendes, E. F. (2023). Machine learning advances for time series forecasting. *Journal of Economic Surveys*, 37(1):76–111.
- Nagy, P., Frey, S., Sapora, S., Li, K., Calinescu, A., Zohren, S., and Foerster, J. (2023). Generative AI for End-to-End Limit Order Book Modelling: A Token-Level Autoregressive

- Generative Model of Message Flow Using a Deep State Space Network. In *ACM Other conferences*, pages 91–99. Association for Computing Machinery, New York, NY, USA.
- Nicholas Refenes, A., Zapranis, A., and Francis, G. (1994). Stock performance modeling using neural networks: A comparative study with regression models. *Neural Networks*, 7(2):375–388.
- Olson, D. and Mossman, C. (2003). Neural network forecasts of Canadian stock returns using accounting ratios. *International Journal of Forecasting*, 19(3):453–465.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.
- Pesaran, M. H. (2015). *Time Series and Panel Data Econometrics*. Oxford Scholarship Online Economics and Finance. Oxford University Press, Oxford, first edition edition.
- Qian, E., Sorensen, E. H., and Hua, R. (2007). Information Horizon, Portfolio Turnover, and Optimal Alpha Models. *The Journal of Portfolio Management*, 34(1):27–40.
- Rahimi, A. and Recht, B. (2008a). Uniform approximation of functions with random bases. In *2008 46th Annual Allerton Conference on Communication, Control, and Computing*, pages 555–561, Monticello, IL, USA. IEEE.
- Rahimi, A. and Recht, B. (2008b). Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc.
- Roncalli, T. (2016). *Introduction to Risk Parity and Budgeting*. Chapman and Hall/CRC, 1st edition.
- Saad, E. W., Prokhorov, D. V., and Wunsch, D. C. (1998). Comparative study of stock trend

- prediction using time delay, recurrent and probabilistic neural networks. *IEEE Trans. Neural Networks*, 9(6):1456–1470.
- Schmidt, W. F., Kraaijveld, M. A., and Duin, R. P. W. (1992). Feedforward neural networks with random weights. In *Proceedings., 11th IAPR International Conference on Pattern Recognition. Vol.II. Conference B: Pattern Recognition Methodology and Systems*, pages 1992–03. IEEE.
- Scornet, E. and Hooker, G. (2026). Theory of Random Forests. *Annual Review of Statistics and Its Application*, 13(Volume 13, 2026):99–121.
- Sharma, M., Sartor, E., Cavazza, M., and Prendinger, H. (2025). Echo State Networks for Bitcoin Time Series Prediction. *Working Paper*, arXiv:2508.05416.
- Shwartz-Ziv, R. and Armon, A. (2022). Tabular data: Deep learning is not all you need. *Information Fusion*, 81:84–90.
- Sirignano, J. and Cont, R. (2019). Universal features of price formation in financial markets: Perspectives from deep learning. *Quantitative Finance*, 19(9):1449–1459.
- Sneddon, L. (2008). The Tortoise and the Hare: Portfolio Dynamics for Active Managers. *The Journal of Investing*, 2008(4):106–111.
- Sreenivasan, K., Sohn, J.-y., Yang, L., Grinde, M., Nagle, A., Wang, H., Xing, E., Lee, K., and Papailiopoulos, D. (2022). Rare Gems: Finding Lottery Tickets at Initialization. *Advances in Neural Information Processing Systems*, 35:14529–14540.
- Sun, C., Song, M., Cai, D., Zhang, B., Hong, S., and Li, H. (2024). A Systematic Review of Echo State Networks From Design to Application. *IEEE Transactions on Artificial Intelligence*, 5(1):23–37.
- Tanaka, G., Yamane, T., Héroux, J. B., Nakane, R., Kanazawa, N., Takeda, S., Numata, H., Nakano, D., and Hirose, A. (2019). Recent advances in physical reservoir computing: A review. *Neural Networks*, 115:100–123.

- Tian, Z. and Pei, L. (2024). Stocks price prediction based on optimized echo state network by sparrow search algorithm. *International Journal of Dynamics and Control*, 12(8):2764–2777.
- Tölö, E. (2020). Predicting systemic financial crises with recurrent neural networks. *Journal of Financial Stability*, 49:100746.
- Trierweiler Ribeiro, G., Alves Portela Santos, A., Cocco Mariani, V., and dos Santos Coelho, L. (2021). Novel hybrid model based on echo state neural network applied to the prediction of stock price return volatility. *Expert Systems with Applications*, 184:115490.
- Vlachas, P. R., Pathak, J., Hunt, B. R., Sapsis, T. P., Girvan, M., Ott, E., and Koumoutsakos, P. (2020). Backpropagation algorithms and Reservoir Computing in Recurrent Neural Networks for the forecasting of complex spatiotemporal dynamics. *Neural Networks*, 126:191–217.
- Vuletić, M., Prenzel, F., and Cucuringu, M. (2024). Fin-GAN: Forecasting and classifying financial time series via generative adversarial networks. *Quantitative Finance*, 24(2):175–199.
- Wang, J. and Chen, Z. (2024). Factor-GAN: Enhancing stock price prediction and factor investment with Generative Adversarial Networks. *PLoS One*, 19(6):e0306094.
- Wang, S., Li, Z., and Li, Q. (2023). Inverse Approximation Theory for Nonlinear Recurrent Neural Networks. In *The Twelfth International Conference on Learning Representations*.
- Yu, P. and Yan, X. (2020). Stock price prediction based on deep neural networks. *Neural Computing and Applications*, 32(6):1609–1628.
- Zhang, G., Eddy Patuwo, B., and Y. Hu, M. (1998). Forecasting with artificial neural networks:: The state of the art. *International Journal of Forecasting*, 14(1):35–62.
- Zhang, H., Liang, J., and Chai, Z. (2013). Stock Prediction Based on Phase Space Reconstruction and Echo State Networks. *Journal of Algorithms & Computational Technology*, 7(1):87–100.

Zhang, T. (2023). *Mathematical Analysis of Machine Learning Algorithms*. Cambridge University Press, Cambridge, England, UK.

Zhao, J., Schaefer, F. T., and Anandkumar, A. (2022). ZerO Initialization: Initializing Neural Networks with only Zeros and Ones. *Transactions on Machine Learning Research*.

APPENDIX

A AlgoSeek Intraday Stock Data

We provide a summary snapshot of the data we have access to for each minute and each traded contract in Table A.1 below:

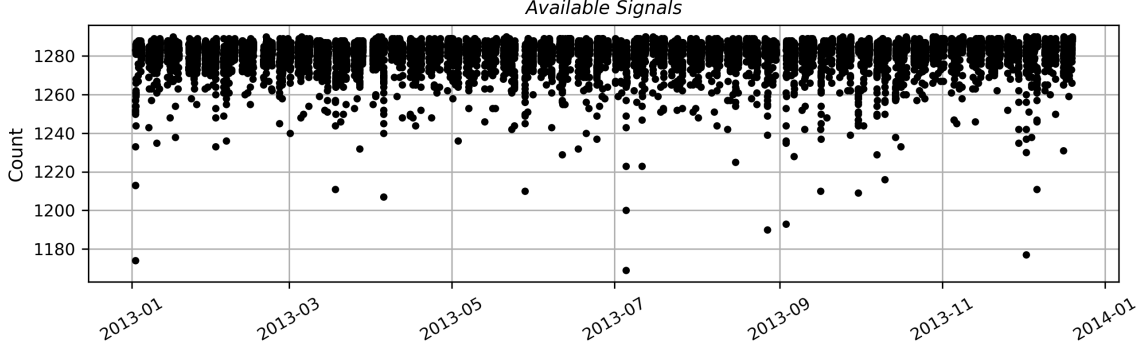
Table A.1: AlgoSeek OHLC bar table example

#	Field	Type	Description
1	Date	YYYYMMDD	Trade Date.
2	Ticker	String	Ticker Symbol.
3	TimeBarStart	HHMM, HHMMSS, HHMMSSMMM	Time stamps are EST. For minute bars, the format is HHMM. For second bars, the format is HHMMSS. Examples: One second bar 130302 is from time greater than 130302.000 to 130302.999. One minute bar 1104 is from time greater than 110400.000 to 110459.999.
4	FirstTradePrice	Number	Price of first trade.
5	HighTradePrice	Number	Price of highest trade.
6	LowTradePrice	Number	Price of lowest trade.
7	LastTradePrice	Number	Price of last trade.
8	VolumeWeightPrice	Number	Trade Volume weighted average price: $Sum((Trade1Shares \times Price) + (Trade2Shares \times Price) + \dots) / TotalShares$
9	Volume	Number	Total number of shares traded.
10	TotalTrades	Number	Total number of trades.

B Signal Construction

The signal construction workflow requires the estimation of market factors to obtain the residual idiosyncratic term $U_t^{(i)}$. We use PCA on the covariance matrix of returns to consistently estimate the factors:

Figure C.1: Signal Availability over the Forecasting Period



- We transform each stock return time series into a z -score, and subsequently calculate the associated pairwise covariance matrix as

$$\rho_{ij} = \frac{1}{T-1} \sum_{t=1}^T L_t^{(i)} L_t^{(j)} \quad \text{and} \quad L_t^{(i)} = \frac{r_t^{(i)} - \bar{r}^{(i)}}{\bar{\sigma}^{(i)}}.$$

- From the correlation matrix, PCA yields a vector of eigenvalues and corresponding eigenvectors v_i^j . For each index j , the eigenportfolio and the eigenportfolio returns are thus

$$Q_i^j = \frac{v_i^j}{\sigma^{(i)}} \quad \text{and} \quad F_{j,t} = \sum_{i=1}^N \frac{v_i^j}{\sigma^{(i)}} r_t^{(i)}, \quad j = 1, 2, \dots, J. \quad (14)$$

We can then proceed as discussed in Section 3.3 to estimate the discrete-time analog of the OU process for $U_t^{(i)}$, which then yields, combining models at different levels of discretization, our vector of stock-specific signals.

C Model and Estimation Details

Missing Data and State Decay. As discussed in Section 3.2, the entire cross-section of returns and signals might not be fully available at all time indices. Indeed, as shown graphically in Figure C.1, although consistently high, the count of observable asset signals changes often over the course of our panel. As such, we design our ESN models in such a way as to be robust to missing data. Our strategy is to introduce *reservoir decay* whenever input signals are missing. Formally, recall that the i th stock state equation can be written as

$$X_t^{(i)} := \alpha X_{t-1}^{(i)} + (1 - \alpha) \varphi \left(A X_{t-1}^{(i)} + C Z_t^{(i)} \right).$$

Table C.1: ESN Training and Cross-validation Parameters

Training Parameter		Horizon				
		10 min	30 min	60 min	2 hours	EOD
$M_{t,h}$	Window size	30 min	30 min	60 min	60 min	1 day
τ_h	Window buffer	10 min	30 min	60 min	2 hours	1 day
	CV frequency	1 day	1 day	1 day	1 day	1 day
	CV window size	1 week	1 week	1 week	1 week	1 week
	CV split ratio	0.7	0.7	0.7	0.7	0.7

Suppose now that the ESN must ingest a signal sequence that includes a **NaN** value,

$$\dots, Z_{t-1}^{(i)}, \text{NaN}, Z_{t+1}^{(i)}, \dots$$

where we use **NaN** to signify missing data, as in the numerical setting. Reservoir decay functionally means that any **NaN** entry in the sequence of input signals is replaced with a zero vector. Therefore, we would obtain states

$$\begin{aligned} X_{t-1}^{(i)} &:= \alpha X_{t-2}^{(i)} + (1 - \alpha) \varphi \left(A X_{t-2}^{(i)} + C Z_{t-1}^{(i)} \right), \\ X_t^{(i)} &:= \alpha X_{t-1}^{(i)} + (1 - \alpha) \varphi \left(A X_{t-1}^{(i)} \right), \\ X_{t+1}^{(i)} &:= \alpha X_t^{(i)} + (1 - \alpha) \varphi \left(A X_t^{(i)} + C Z_{t+1}^{(i)} \right). \end{aligned}$$

So long as $\alpha \in [0, 1)$ and the spectral radius of A is below unity, this approach allows treating sequences with arbitrary sub-sequences of missing data: The state equation is validly iterated forward, and the states decay (contract) towards the zero vector. This enables us not to “throw away” our previously collected state sequence. Nonetheless, to conform with the setup of Section 3.2, any state that is associated with state decay induced by missing signal observation (or missing returns) is not included at the time of estimation for the output coefficients.

Training Windows and Buffers. The training procedure outlined in Section 4.2 requires the choice of some parameters, such as training window size and buffer, as well as cross-validation frequency, window size, and ratio. We collect our choices for these quantities in Table C.1.

For window size, we compromise between the length of the window and the negative effect that including too much past data can have on predictive power. For the window buffer, we always choose the minimal length possible given the prediction horizon. Cross-validation to select the diagonal, anisotropic ridge penalty matrix $\Lambda_{t,h}$, is run every trading day with one week of past data. We keep a constant split ratio of 70% training and 30% validation across all horizons.

ESN Hyperparameter Optimization. To search for the horizon-specific ESN hyperparameters detailed in Table 1, we rely on the Python model optimization library Optuna (Akiba et al., 2019). Our tuning sample includes only data from September to December 2012. We keep the same training and cross-validation parameters outlined in Table C.1, and set as the objective to minimize the cumulated MSFE.

Random Forest Hyperparameters. We use `RandomForestRegressor` from `scikit-learn`. The following hyperparameters are set for all horizons: `min_samples_split=10`, `max_depth=10`, and `n_estimators=100`. To ensure that the RF is grown by bootstrapping trees also across different subsets of signals in $Z_t^{(i)}$, we set `max_features='sqrt'`. Finally, the option `n_jobs=-1` is set to allow for parallelization across all available CPU cores.

D Details on Portfolio Construction

Asset Selection. We construct a dense subset of assets based on volume at the 10-minute interval and 10-minute horizon, which in our setting is the only combination that ensures non-overlapping returns. We enforce that all assets included in the final panel used for portfolio construction and evaluation are such that both signals and returns are available at all 10-minute observation timestamps. This selection thus removes thinly traded or illiquid assets that may not be traded at high frequency. The final fixed cross-section size is $\overline{N} = 388$, which is still an order of magnitude larger than the number of daily observations.

Covariance estimation. To estimate the previous day’s covariance estimate, $\hat{\Sigma}_{[t]}$, we use the nonlinear shrinkage estimator of [Ledoit and Wolf \(2004\)](#) as implemented by the `scikit-learn` routine `LedoitWolf` with default block size. This estimate is then used in the construction of all forecast-based MV portfolio weights to ensure comparability across methods.

Figure D.1: Portfolio Weights

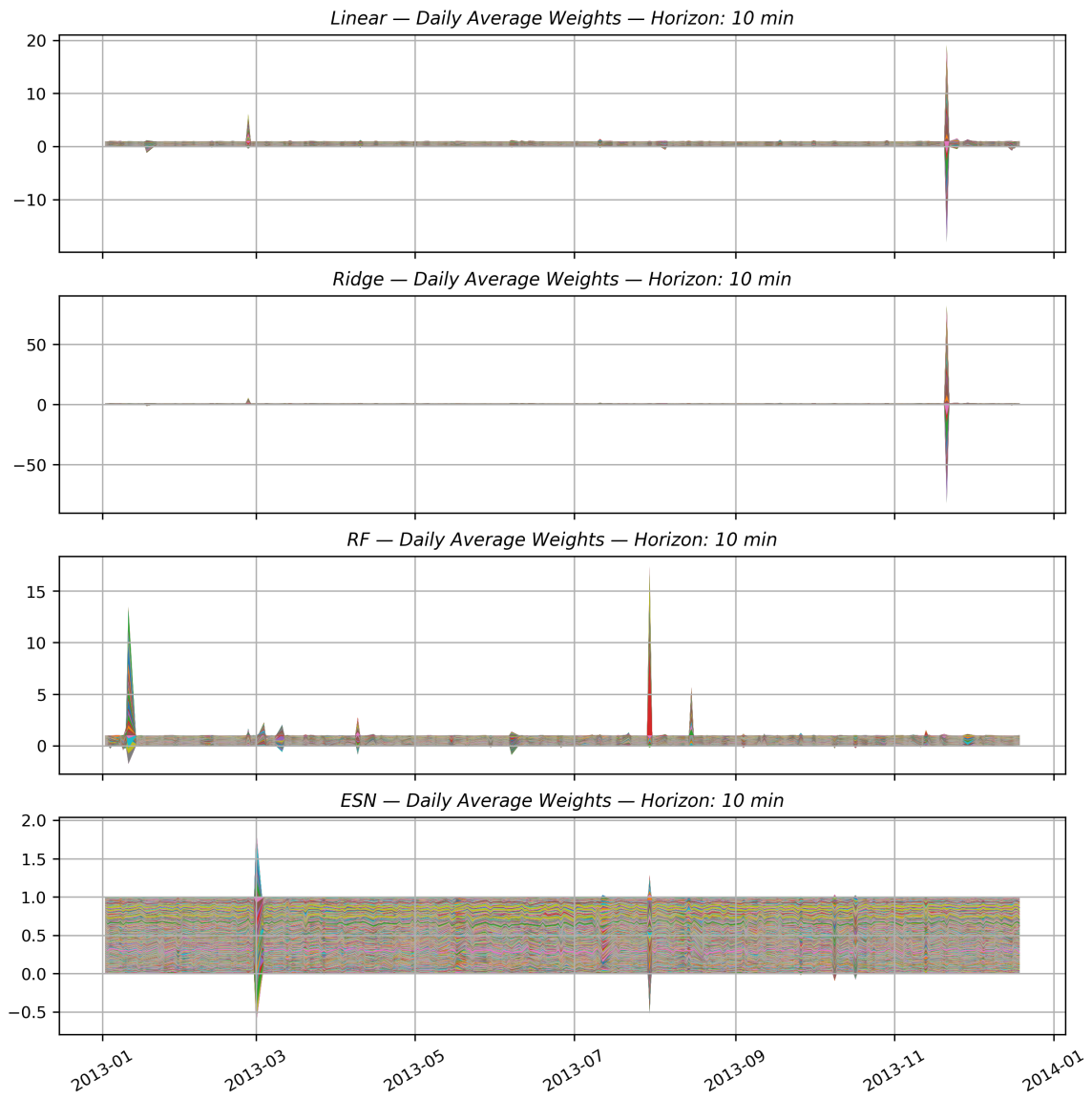
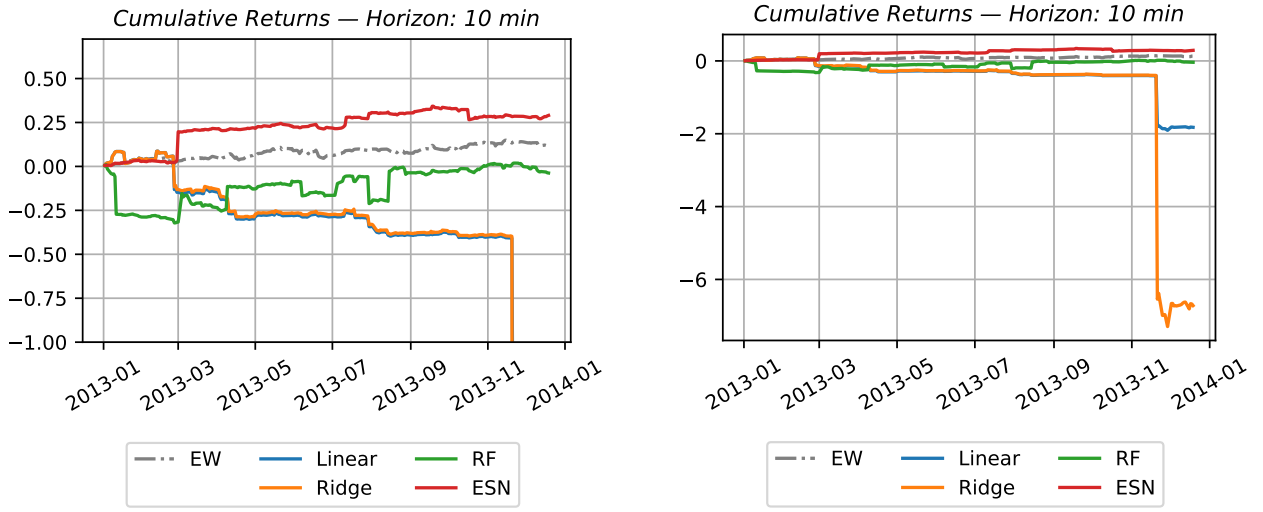
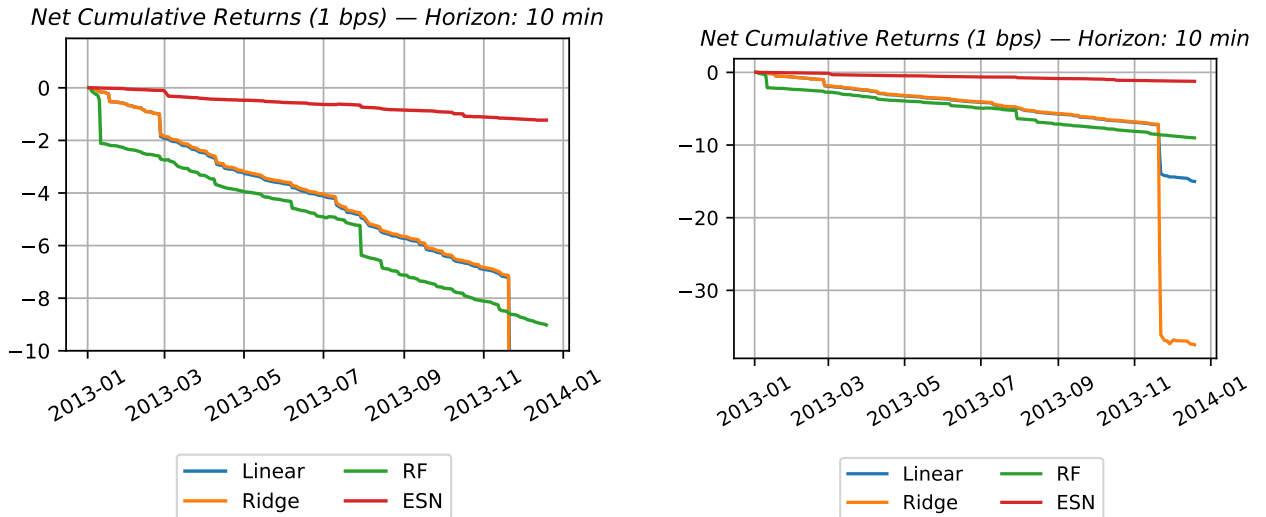


Figure D.2: Cumulative Portfolio Returns



Notes: Portfolio weights derived with linear, ridge, random forest, and ESN forecasts are computed on identical daily covariance matrix estimates with Ledoit-Wolf nonlinear shrinkage. Cumulative returns are net of transaction costs. Left panel uses a reduced y -axis, while the right panel is unrestricted.

Figure D.3: Net Cumulative Portfolio Returns



Notes: Net returns for all portfolios computed with 1bps transaction cost from cumulative returns and turnovers, cf. Figure 5. Left panel uses a reduced y -axis, while the right panel is unrestricted.