International Journal of Approximate Reasoning 132 (2021) 70–85

Contents lists available at ScienceDirect

International Journal of Approximate Reasoning

www.elsevier.com/locate/ijar

# Spatiotemporal adaptive neural network for long-term forecasting of financial time series

Philippe Chatigny [a], Jean-Marc Patenaude [b], Shengrui Wang [a,*]

[a] Department of Computer Science, Université de Sherbrooke, Sherbrooke, Québec, Canada
[b] Laplace Insights, Sherbrooke, Québec Canada

ARTICLE INFO

*Article history:*
Received 9 June 2020
Received in revised form 16 November 2020
Accepted 2 December 2020
Available online 8 December 2020

*Keywords:*
Time series forecasting
Semi-supervised learning
Dynamic factor graphs
Neural networks

ABSTRACT

Optimal decision-making in social settings is often based on forecasts from time series (TS) data. Recently, several approaches using deep neural networks (DNNs) such as recurrent neural networks (RNNs) have been introduced for TS forecasting and have shown promising results. However, the applicability of these approaches is being questioned for TS settings where there is a lack of quality training data and where the TS to forecast exhibit complex behaviors. Examples of such settings include financial TS forecasting, where producing accurate and consistent long-term forecasts is notoriously difficult. In this work, we investigate whether DNN-based models can be used to forecast these TS conjointly by learning a joint representation of the series instead of computing the forecast from the raw time-series representations. To this end, we make use of the dynamic factor graph (DFG) to build a multivariate autoregressive model. We investigate a common limitation of RNNs that rely on the DFG framework and propose a novel variable-length attention-based mechanism (ACTM) to address it. With ACTM, it is possible to vary the autoregressive order of a TS model over time and model a larger set of probability distributions than with previous approaches. Using this mechanism, we propose a self-supervised DNN architecture for multivariate TS forecasting that learns and takes advantage of the relationships between them. We test our model on two datasets covering 19 years of investment fund activities. Our experimental results show that the proposed approach significantly outperforms typical DNN-based and statistical models at forecasting the 21-day price trajectory. We point out how improving forecasting accuracy and knowing which forecaster to use can improve the excess return of autonomous trading strategies.



## 1. Introduction

In recent decades, DNNs have improved TS forecast accuracy in various social settings [1]. Besides their ability to handle non-linear processes, they provide a cost-effective approach for uncovering relations between TS. DNNs are based on the dynamic factor graph (DFG) framework [2,3], which is a particular case of a factor graph [4] in which the template method [4] is applied. Specifically, a DNN-based model assumes that the factors of a DFG are individual neural networks (NN) that enforce a hierarchical structure for pattern detectors throughout its hidden layers [5]. Under this framework, the DNN-based

* Corresponding author.
*E-mail addresses:* Philippe.Chatigny@usherbrooke.ca (P. Chatigny), jeanmarc@laplaceinsights.com (J.-M. Patenaude), Shengrui.Wang@usherbrooke.ca (S. Wang).

https://doi.org/10.1016/j.ijar.2020.12.002
0888-613X/

model learns complex probability distributions and increases forecast accuracy on mostly homogeneous datasets containing multiple measurements, as well as in applications where there are exogenous variables that are strongly related to the variable(s) of interest [6]: e.g., traffic [7] or electricity load forecasting [8].

However, training a DNN remains difficult for most TS settings [9,1], especially when TS are non-ergodic, heteroskedastic, non-stationary or have high noise-to-signal ratios. Such cases are often found in financial TS. Few DNN-based models have demonstrated consistent accuracy on such datasets spanning multiple years for different asset classes [10]. Besides reasons associated with concept drift [11], the difficulties in forecasting financial TS are also due to the fact that most DNN learning frameworks do not appear to be adapted for this setting. Training a DNN needs a large dataset of independent training samples that are representative of the data to infer. Aside from applications like intra-day forecasting [12], most financial applications rely on TS that have a relatively limited number of measurements [9]. Additionally, historical price trajectories can be very noisy and their behaviors exhibit complex cyclical effects [13]. As it is not possible to obtain multiple independent realizations of a specific asset's price fluctuation under different circumstances for the same time period [14], the nature of financial TS necessarily leads to both a lack of training data and the well-known difficulty in modeling their long-term effects [15].

This paper proposes a more effective DNN framework for forecasting multiple financial assets conjointly and enhancing the capability of the TS model to learn a larger set of probability distributions. The key contributions of this paper are as follows:

1. We propose a novel attention mechanism for the Dynamic Factor Graph (DFG) framework. This mechanism offers the capacity to consider a variable number of past latent states over time.
2. We make use of the novel attention mechanism to optimize the order of an autoregressive (AR) generative function over time. We show how such a mechanism can model non-stationary distributions while keeping a constant parameterization.
3. By incorporating the attention mechanism, we develop an energy-based deep generative approach for modeling interactions between multiple TS to generate multivariate forecasts. Our spatiotemporal adaptive neural network (STANN) is able to operate under a limited data constraint by exploiting prior knowledge of the TS to "virtually" augment its training samples and allows the discovery of interrelations between TS.
4. We have conducted an extensive experimental evaluation showing the effectiveness of the proposed model for forecasting 21 daily return trajectories of exchange-traded funds (ETFs) and mutual funds (MFs). We also show preliminary but promising results of the proposed model for improving autonomous trading strategies. Of all the models proposed in the last 10 years [10], ours is, to our knowledge, the first to outperform naive baselines in a monthly multivariate financial TS setting.

The remainder of this paper is organized as follows: Section 2 reviews major existing work on modeling TS in social settings and relevant notions related to the DFG. In Section 3, we present our model and describe its training procedure. In Section 4, we present the setup of our empirical evaluation, which extends over more than 19 years of financial market activities, and describe our results. Section 5 presents our conclusion.

## 2. Related work

### 2.1. Prior work

Different formulations [16,17] of DNN models have been introduced to facilitate their application on TS data. While promising results have been achieved recently for financial TS prediction, as in [18], it has been pointed out that much of the published machine learning (ML) work in the TS literature claims satisfactory accuracy without adequately comparing the methods used against conventional methods [9] and, further, that it relies on inappropriate criteria [19,20]. In fact, only a few authors, such as [21,22], have been able to show that their models yield better performance on multiple TS than simple statistical models like ARIMA or even a naive forecast. Most work uses non-scaled error metrics to assess forecast quality on multiple TS. However, it has been known for years that comparing forecasts of multiple TS of different scales via non-scaled metrics often leads to misleading results [19,20]. The myriad of existing DNN-based models [10] applied to financial settings and the results presented around them have raised undue expectations that such methodologies provide accurate predictions at forecasting multiple TS, while there is clearly a lack of experimental demonstration that they outperform simple baselines in the majority of cases.

Nonetheless, large gains can still be achieved by using DNN and ML approaches. Recently, state-of-the-art accuracy was achieved at the M4 competition [23], where the top 2 entries used DNN-based or ML techniques along with statistical models. Subsequent to these findings, the authors in [24] were the first to show that it was possible to build a pure DNN-based model for this task and achieve greater gains than the best competition entry [22]. Given the wide range of TS to forecast,[1] the top-performing models submitted relied on ensemble techniques to be robust over the different types of series.

[1] The M4 dataset contained 100,000 individual TS, of which approximately 25% were financial TS of different types.

Direct comparison between single and ensemble models is generally unfair, as ensemble models permit the modeling of various probability distributions using multiple TS models and subsequently apply some form of forecast combination by evaluating the inference capabilities of each TS model a posteriori. However, the findings from these models can be investigated with a view to building better individual models. For instance, the DNN-based models, which performed well on this dataset [22,24], provide insights into techniques that can be used to improve the performance of individual models: e.g., residual connections between hidden layers, adaptive learning rate scheduling, input preprocessing and both seasonal and trend decomposition embedded directly in the model. Most of these techniques are *"tricks"* to facilitate DNN learning. However, the idea of applying a signal decomposition within a neural network is promising and several authors [24–26] have shown its effectiveness on real-world datasets. Given the well-known difficulty of dealing with the raw signal of financial TS, we raise the question whether a better representation of these TS can be learned directly by applying such decomposition within the learned latent variables of a DFG [2].

### 2.2. Dynamic factor graph

A DFG consists of an undirected acyclic state-space model where factors are replicated on a fixed time interval $T = \{t_1, ..., t_T\}$ to model a probability distribution. A DFG models the joint probability $P(X, Z; \mathbf{W})$ of the observable values $X = \{x_1, ..., x_T\}$ and the latent variables $Z = \{z_1, ..., z_T\}$ given some parameterization of all factors in the graph $\mathbf{W}$ as in Eq. (1). Here, $\mathscr{L}$ is the partition function and $E(X, Z; \mathbf{W}) \propto -log P(X, Z|W) + const$ is the *total energy* of the model. The *total energy* of the model is the sum of the normalized probability scalars assigned by a factor to all possible input data points associated with it. To obtain a probability, the total energy is normalized by $\mathscr{L}$.

$$P(X, Z; \mathbf{W}) = \frac{e^{-\beta E(X,Z;\mathbf{W})}}{\int_{X'} \int_{Z'} e^{-\beta E(x',z';\mathbf{W})} dx' dz'} = \frac{e^{-\beta E(X,Z;\mathbf{W})}}{\mathscr{L}} \tag{1}$$

$$E(X, Z; \mathbf{W}) = \sum_{t \in T} \sum_{F \in \mathscr{F}} E(A_t, O_t; F);\ A_t \in Z,\ O_t \in \{X, Z\} \tag{2}$$

$$E(A_t, O_t; F) = \begin{cases} error(g(Z_t, \mathbf{W}_g), Z_{t+1}) & \text{if } F = g \\ error(d(Z_t, \mathbf{W}_d), X_t) & \text{if } F = d \end{cases};\ \mathscr{F} = \{d, g\} \tag{3}$$

The *energy term* for a given sequence of observable values $X$ and latent states $Z$ is given by Eq. (2), with the energy term of a single factor defined by Eq. (3). Here we assume that our DFG follows a parameterization similar to that of an HMM architecture of order 1 with two factors, i.e., $\mathscr{F} = \{d, g\}$, and $\mathbf{W}_F$ is the parameterization of the factor $F \in \mathscr{F}$. The higher the energy term between an input data point and its associated output data point, the less probable it is that the value will be observed. Despite the fact that the DFG's edges are undirected, the *energy term* of each factor is not. Hence, training a DFG for TS forecasting is similar to adjusting the parameters of a Dynamic Bayesian Network (DBN) [4] where we simply need to adjust the parameters of the factors using maximum likelihood estimation, which is equivalent to reducing the *total energy* of the model.

For our particular case, where we consider an HMM under the DFG framework, as illustrated in Fig. 1, the main difference between an RNN and this particular DFG is how the state-space component is used. However, since Eq. (1) is intractable for continuous variables under non-Gaussian distributions, we estimate the mode of the distribution instead by maximum a posteriori approximation [3]. Thus, an HMM-based DFG model learns the probability distribution $\mathbb{P}$ of a TS using two factors replicated over time: a decoder factor and a dynamic factor, i.e. $\mathscr{F} = \{d, g\}$. The decoder factor $d(Z; \mathbf{W}_d)$ is a function that models the maximum likelihood of observing a random variable $X_t$ given latent variable $Z_t$:

$$\tilde{X}_t = d(Z_t, \mathbf{W}_d) \hat{=} \arg\max_x \mathcal{L}(x|Z_t = z_i; \mathbf{W}_d);\ z_i \in Z, x \in X \tag{4}$$

with $\mathbf{W}_d$ being the parameterization of the factor, $t$ a particular time point and $\mathcal{L}$ the likelihood function. The dynamic factor $g(Z; \mathbf{W}_g)$ models the maximum likelihood of observing a state given some prior state and is defined by Eq. (5). $g(Z; \mathbf{W}_g)$ models a transition probability distribution as in the DBN framework:

$$Z_{t+1} = g(Z_t, \mathbf{W}_g) \hat{=} \arg\max_z \mathcal{L}(Z_{t+1} = z|Z_t = z_j, \cdots, Z_{t-k} = z_{j'});\ z, z_j, z_{j'} \in Z \tag{5}$$

Note that one must specify the order of $g(Z; \mathbf{W}_g)$ by changing its configuration: $Z_{t+1} = g(Z_t, ..., Z_{t-k}; \mathbf{W}_g))$, where $k$ is the autoregressive (AR) order of the process. Doing so makes the assumption that the probability distribution $\mathbb{P}$ models a stationary process if, for all $t$, $\mathcal{L}(Z_{t+1} = z_i|Z_{t-k:t} = z_j) \geq 0$ and is constant for all $t$ [4].[2] This assumption holds for both the discrete and the continuous case [27,28]. When modeling $\mathbb{P}$ using a graphical model, we use the notation $\mathbb{P} \models (A \perp\!\!\!\perp B|C)$ [4]

[2] Here, $Z_{t-k:t}$ corresponds to all $Z_{t'}$ included between $Z_{t-k}$ and $Z_t$, where $t' \in T$.

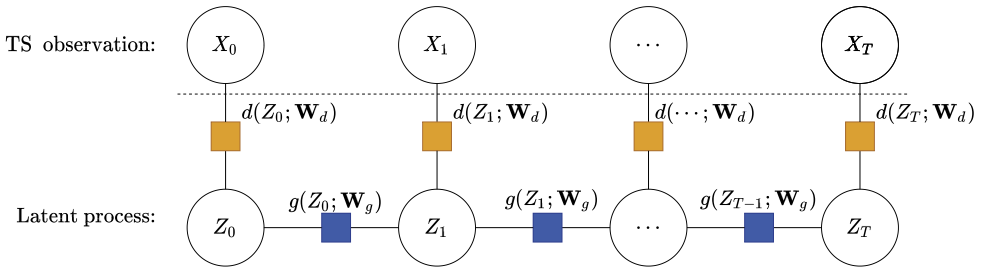


**Fig. 1.** An HMM-based DFG architecture that admits observed variables $X$ and latent variable $Z$. Both decoder (orange squares) and dynamic factors (blue squares) can be implemented as parametric functions and be trained using gradient descent. Notice that the dynamic process of the series is captured entirely in the latent space: $Z_{t+1} = g(Z_t; \mathbf{W}_g)$. Thus, an HMM-based DFG is a particular case of an RNN where the hidden states are directly learned instead of being computed explicitly by a function of past inputs. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

to indicate that $\mathbb{P}$ models a local independence relation between a set of nodes $A$ and $B$ given $C$. In particular, assuming that the probability distribution models a stationary process induces the following set of independence relations:

- The latent variable evolves in a *Markovian* or a *semi-Markovian* way:

$$\mathbb{P} \models (Z_{t+1} \perp\!\!\!\perp Z_{0:t-1-k} | Z_{t-k:t}) \tag{6}$$

- The observation variables at time $t$ are conditionally independent of the state sequence given the latest $k+1$ state variables at time $t$ [4]:

$$\mathbb{P} \models (X_t \perp\!\!\!\perp Z_{0:t-1-k}, Z_{0:t+1:\infty} | Z_{t-k:t}) \tag{7}$$

This AR order is a hyperparameter that needs to be tuned carefully, since the true probability distribution is intractable in most cases [3] and the set of local independences cannot be verified in practice. Assuming that the AR parameters are constant can impair training, as the resulting AR weights are optimized to reduce the average error. This limitation is problematic if the AR order was not selected appropriately or the training data contains multiple TS dynamics. In this work, we address these limitations by proposing an attention mechanism that enables a DFG to select its AR order automatically and adjust it over time. The stationary assumption can thus be relaxed such that the set of interdependences in Eq. (6) and Eq. (7) holds but the process order $k$ is a function of time. This permits non-stationary probability distributions to be modeled since $\mathcal{L}(Z_{t+1} = z_i | Z_{t-k:t}; \mathbf{W}_g) \geq 0$ but is not necessarily constant over time.

## 3. The STANN model

### 3.1. Model definition without including time series dependencies

Given $X : \mathbb{R}^{T \times n \times m}$, a 3-dimensional tensor representing a set of $n$ TS of length $T$ and dimensionality $m$, we define $X_{t,i,j}$ as the value of dimension $j$ for TS $i$ at time $t$. The task of interest is to predict $n$ multivariate TS $\tau$ time steps ahead $\tilde{X} : \mathbb{R}^{\tau \times n \times m}$. We represent the spatial relationship between series within a 3-dimensional tensor, that we denoted by $W : \mathbb{R}^{n \times R \times n}$, where $R$ is the number of relations considered. Thus, our aim is to train a model $f : \mathbb{R}^{T \times n \times m} + [\mathbb{R}^{n \times R \times n}] \rightarrow \mathbb{R}^{\tau \times n \times m}$

In STANN, we use a particular formulation of the DFG. The *decoder* factor $d(Z; \mathbf{W}_d)$ decodes the expected variation between $X_{t-1}$ and $X_t$ from the latent factor $Z_t$, which allows the decoder to be defined as in Eq. (8). Here $\tilde{X}_t$ is the prediction computed at time $t$.

$$\tilde{X}_t = X_{t-1} + d(Z_t; \mathbf{W}_d) \tag{8}$$

The dynamical module $g(Z; \mathbf{W}_g)$ is defined by Eq. (9) and considers the past $k+1$ relevant latent factors $Z_{t-k:t}$, i.e. $Z_{t-k}$ to $Z_t$. $d(Z; \mathbf{W}_d)$ and $g(Z; \mathbf{W}_g)$ is implemented as a doubly residual stacking NN, as in N-BEATS [24]. In contrast to N-BEATS, we apply the TS decomposition on the latent factors rather than the raw signals.

$$\tilde{Z}_{t+1} = g(Z_{t-k:t}; \mathbf{W}_g) \tag{9}$$

### 3.2. Adaptive computation time for autoregressive order selection

As mentioned in the previous section, assuming that the forecast depends on a fixed AR order covering the past $k$ observations is a strong assumption that can impair model training if the autoregressive order is not selected correctly. RNNs, like the long short-term memory (LSTM) network [29], consider the past $k$ observations by maintaining in memory

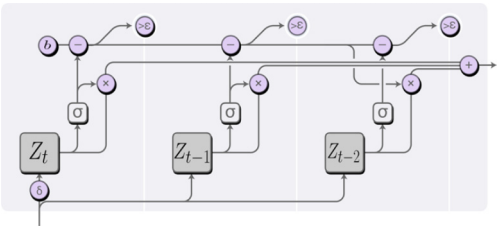


**Fig. 2.** Illustration of the proposed attention mechanism. For illustration purposes, $b$ includes both $b_{\texttt{time}}$ and $b_{\texttt{cost}}$, and $\sigma$ denotes the AR weight produced by $actm(Z_{t-i})$. The drawing was adapted from [31].

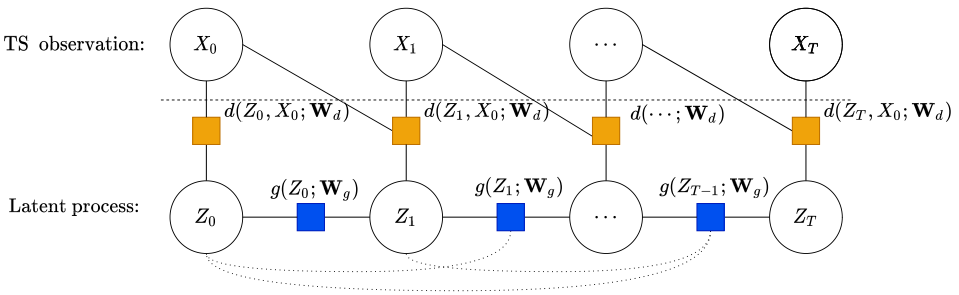


**Fig. 3.** Illustration of our model. The dotted lines at the bottom of the graph represent the possible relations between variables and the dynamic factors that $actm(Z; \mathbf{W}_{actm})$ can consider. For illustration purposes, $actm(Z; \mathbf{W}_{actm})$ and $g(Z; \mathbf{W}_g)$ are represented by the same factor (blue square in this figure).

a state vector that allows them to retain information as long as required and forget it when it is no longer relevant. Unlike LSTM, we permit $k$ to vary adaptively without requiring the observable value as input. To this end, we propose an adaptive attention-based mechanism to enable DFG to be *memory-augmented*. Our attention mechanism is inspired by the *Adaptive Computation Time* (ACT) algorithm proposed in [30], denoted as $actm(Z; \mathbf{W}_{actm})$ in our model. $actm(Z; \mathbf{W}_{actm})$ is a factor that generates a probability distribution on $Z$ that is used to select the order of the regression in $Z_{t-k:t}$. The probability function associated with the factor at each time $t$ is modeled by a parametric function $f_{actm}(Z_t; \mathbf{W}_{actm})$, with $\mathbf{W}_{actm}$ being its parameterization. For ease of computation, the order selection and the computation of $Z_{t-k:t}$ are performed by using the sum of a certain number of past latent factors weighted by probability distribution function $f_{actm}$ as in Eq. (10a).

$$Z_{t-k:t} = actm(Z_t, \mathbf{W}_{actm}) = \sum_{0 \le k' < k:[b(t) > \epsilon]} \varphi_{Z_{t-k'}} Z_{t-k'} \tag{10a}$$

$$\varphi_{Z_{t-k'}} = \begin{cases} f_{actm}(Z_{t-k'}, \mathbf{W}_{actm}) & \text{if } b(t) - f_{actm}(Z_{t-k'}, \mathbf{W}_{actm}) > \epsilon \\ b_{\texttt{cost}} & \text{if } b(t) - f_{actm}(Z_{t-k'}, \mathbf{W}_{actm}) \le \epsilon \end{cases} \tag{10b}$$

Specifically, $actm(Z; \mathbf{W}_{actm})$ is implemented with two budgets $b(t) = \{b_{\texttt{time}} = t, b_{\texttt{cost}} = 1\}$: one to keep account of available past time steps and one to track the cost of considering a latent factor. We start with the current latent state factor $Z_t$ and consider whether to include $Z_{t-k'}$ for $k' = 0, 1, 2, \ldots$. Each time we consider a latent factor $Z_{t-k'}$, we reduce our budget $b_{\texttt{time}}$ by 1 and $b_{\texttt{cost}}$ by $\varphi_{Z_{t-k'}} = f_{actm}(Z_{t-k'}, \mathbf{W}_{actm})$, the latter being bounded within $]0, 1[$. If a budget goes below $\epsilon$, i.e., either $b_{\texttt{cost}} < \kappa$ or $b_{\texttt{time}} = 0$, we stop considering any more latent factors and attribute the remaining cost budget to the last factor considered. $\kappa \in \mathbb{R}^+$ is a small constant (0.01 for the experiments in this paper), whose purpose is to allow the selection of an AR(1) process.

Hence Eq. (9) can be reformulated as Eq. (11).

$$\tilde{Z}_{t+1} = g(Z_{t-k:t}; \mathbf{W}_g, \mathbf{W}_{actm}) \tag{11}$$

We can interpret $actm(Z; \mathbf{W}_{actm})$'s objective as evaluating the quality of each past latent factor and assigning the appropriate autoregressive weight at times $t - k'$ that maximizes the log likelihood of the generative process modeled by Eq. (9). Since $actm(Z; \mathbf{W}_{actm})$ uses $b_{\texttt{time}}$ to determine how many past steps are available, we can theoretically account for all previous learned factors if $\sum_{k=0}^{t} f_{actm}(Z_{t-k}, \mathbf{W}_{actm}) < 1 - \kappa$. Note that the imposed budget restricts each autoregressive weight to be between 0 and 1, with the sums of all the weights being equal to 1. We apply this mechanism solely within $g(Z; \mathbf{W}_g)$ to facilitate the training model, but the approach could also be extended to $d(Z; \mathbf{W}_d)$. From now on, then, we will simplify our notation by considering that $\mathbf{W}_g$ also includes the ACTM parameters. The attention mechanism is summarized in Fig. 2 and can be designed as any configuration of a feedforward network with a sigmoid activation function. An illustration of our model with ACTM is presented in Fig. 3.

The training procedure consists of minimizing the following bi-objectives loss function (12):

$$Loss(d, g, Z) = +\frac{1}{T}\sum_{t=1}^{T-1}\Delta(X_{t-1} + d(Z_t; \mathbf{W}_d), X_t) \tag{12a}$$

$$+\lambda\frac{1}{T}\sum_{t=1}^{T-1}||Z_{t+1} - g(Z_{t-k:t}; \mathbf{W}_g))||^2 \tag{12b}$$

The first term (12a) measures the ability of the model to reconstruct $X_t$ from $Z_t$. The second term (12b) measures the system's capacity to capture the dynamicity of the equation by its ability to link states of $Z$ in sequential order. $\Delta$ is a loss function that measures the difference between the prediction $\tilde{X}_t$ and the ground truth $X_t$. The second term in Eq. (12b) forces the model to learn latent factors $Z_{t+1}$ that are as close as possible to $g(Z_{t-k:t}, \mathbf{W}_g)$, with the ideal case being a solution where $Z_{t+1} = g(Z_{t-k:t}, \mathbf{W}_g)$. However, this solution is not valid for the first term in Eq. (12a), where the ideal case is a solution for which $X_{t+1} = d(Z_{t-k:t}, \mathbf{W}_d)$. To balance the relative importance of the two terms, the hyperparameter $\lambda$ is introduced to reduce or increase the importance of the second term relative to the first term in Eq. (12a). Training the model using Eq. 12[3] can be accomplished using any expectation-minimization-based approach [2] or an end-to-end [32] approach that trains the three factors conjointly.

### 3.3. Model definition including time series dependencies

Let us now introduce the way interrelations between TS are captured. As pointed out in [33], multiple types of relations between financial TS have been uncovered. To test whether this prior knowledge has predictive capability, we propose that the relationships between the dynamic processes of multiple TS be given as additional prior inputs $W \in \mathbb{R}_+^{n\times R\times n}$ to the model, as in [34]. We will first formalize how relationships between series are incorporated into the model and how this allows us, "virtually", to have a high number of training samples. Then, we will describe two extensions of this approach. The first extension allows the strength of these relations to be weighted, and the second allows the model to learn these relations directly without any prior information.

Relationships between the dynamic processes of $n$ TS are incorporated via a tensor $W \in \mathbb{R}_+^{n\times R\times n}$, where $R$ is the number of relation types given as prior. In the following discussion, each sequence is indexed by $X_{t,i}$, while the corresponding hidden state is represented by $Z_{t,i}$. $X_{t,i}$ correspond to the particular observation of the $i^{\text{iem}}$ TS at time $t$ and $Z_{t,i}$ is its corresponding hidden state. We formulate that, at time $t$, $Z_{t+1,i}$ depends on its own latent representation (intradependency) and on the representations of other series (interdependency).

Intradependency is modeled through a linear mapping $\Theta^{(0)} \in \mathbb{R}^{n\times n}$. Interdependency is modeled with one transition matrix $\Theta^{(r)} \in \mathbb{R}^{n\times n}$ for each possible type of relation $r \in R$. $\Theta^{(r)}$ learns the relationship between each TS by applying a linear combination between neighboring TS and we denote $\Theta_i^{(r)}$ as the linear combination learned for relation $r$ of the $i^{\text{iem}}$ series. We denote all the transition matrices by $\Theta^{(R)} \in \mathbb{R}^{R\times n\times n}$ and $W_i^{(r)}$ as the relation given as prior between the $i^{\text{iem}}$ series and other neighboring TS. To evaluate $Z_{t+1}$, we compute the matrix product between the latent space $Z_t$ and its dependencies $(\Theta^{(0)}, \Theta^{(R)})$ as in Eq. (13). The decoder follows along, using $Z_{t,i}$ as inputs, and computes the expected variation as in Eq. (14). $h_g, h_d$ are the respective activation functions of $g(Z; \mathbf{W}_g)$ and $d(Z; \mathbf{W}_g)$.

$$\begin{aligned}\tilde{Z}_{t+1,i} &= g(Z_{t-k:t,i}; \mathbf{W}_g, \Theta) \\ &= h_g(actm_g(Z_t\Theta_i^{(0)} + \sum_{r\in R} W_i^{(r)} Z_t\Theta_i^{(r)}); \mathbf{W}_g)\end{aligned} \tag{13}$$

$$\tilde{X}_{t+1,i} = X_{t,i} + d(Z_{t,i}; \mathbf{W}_d) = h_d(Z_{t,i}; \mathbf{W}_d) \tag{14}$$

Note that $Z_t$ is shared among all series with respect to $g(Z; \mathbf{W}_g)$, but the representation of each series is disentangled explicitly by means of $W$; i.e., $d(Z; \mathbf{W}_d)$ takes as input $Z_{t,i}$, the hidden factor of the $i$th TS. Doing this has two advantages: (1) $g(Z; \mathbf{W}_g)$ can forecast $\tilde{Z}_{t+1}$ with fewer regressors. (2) It "virtually" increases the number of training samples, as we can use time and positional coordinates to make $T \times n$ fixed-size training samples instead of handling TS as sequential data. With respect to each disentangled latent state, the correlation existing between the latent states of two TS would indicate that our model estimates that the TS follows similar trajectories despite $W$ specifying that they are or are not correlated which is possible in part due to $\Theta^{(R)}$.

### 3.4. Model extensions

The two possible extensions proposed in [34] can also be applied to our model. We summarize the extensions here; readers are invited to refer to the original paper [34] for a more detailed explanation. The first extension, denoted by

[3] In our experiments, $\lambda$ was fixed through a hyperparameter optimization that search the optimal value in the following interval [0.01 and 1.0].

**Table 1**
Datasets for experimental evaluation.

| Dataset | $T$ | $n$ | Data type | Time horizon | $\tau$ | # Runs per model |
|---|---|---|---|---|---|---|
| $\mathscr{D}_1$ | 2186 | 10 | daily adj. close | 1996/07/08 - 2007/08/22 | 21 | 100 |
| $\mathscr{D}_2$ | 2000 | 69 | daily adj. close | 2011/05/31 - 2019/05/10 | 21 | 54 |

$T$ is the total number of time points, $n$ the number of series, $\tau$ the number of steps ahead to forecast and # Runs the total number of evaluation runs made. For all datasets, we considered only the closing price ($m = 1$).

`STANN-R`, consists of adding a learned matrix of weights $\Gamma^r \in \mathbb{R}_+^{n\times n}$ that can reduce the strength of relations given as prior. The second extension, denoted by `STANN-D`, consists of replacing $W$ with $\Gamma$ such that the model learns both the relational structure and the relation weights within $\Gamma$. Applying the `STANN-R` or `STANN-D` extension formalizes Eq. (13) as in Eq. (15) or Eq. (16), respectively, where $\odot$ signifies element-wise multiplication between two matrices:

$$Z_{t+1,i} = g(Z_{t-k:t,i}; \mathbf{W}_g, \Theta) = g(actm_g(Z_t\Theta_i^{(0)} + \sum_{r\in R}(\Gamma_i^{(r)} \odot W_i^{(r)})Z_t\Theta_i^{(r)}) \tag{15}$$

$$Z_{t+1,i} = g(Z_{t-k:t,i}; \mathbf{W}_g, \Theta) = g(actm_g(Z_t\Theta_i^{(0)} + \sum_{r\in R}\Gamma_i^{(r)}\Theta_i^{(r)}) \tag{16}$$

The optimization problem can thus be adjusted for $\Gamma$, depending on whether the dynamic function is specified by Eq. (15) or Eq. (16), and can be written as Eq. (17). $|\Gamma|$ is a $l_1$ regularizing term intended to sparsify $\Gamma^{(r)}$; $\gamma$ is a hyperparameter set to tune this term; and $\lambda$ is a factor set to balance the relative importance of $g(Z; \mathbf{W}_g)$ and $d(Z; \mathbf{W}_d)$.

$$\begin{aligned} d^*g^*, actm_g^*, \Theta^*, \Gamma^* = \underset{d,Z,\Gamma,}{\operatorname{argmin}} \frac{1}{T}\sum_t \Delta(d(Z_t; \mathbf{W}_d) + X_{t-1}, X_t) \\ +\gamma|\Gamma| + \lambda\frac{1}{T}\sum_{t=1}^{T-1}||Z_{t+1} - g(Z_{t-k:t}; \mathbf{W}_g, \Theta)||^2 \end{aligned} \tag{17}$$

## 4. Experiments

### 4.1. Datasets and experimentation procedure

We report here the results of an experimental evaluation of our forecasting methods on two datasets: $\mathscr{D}_1 =$ `Fasttrack` and $\mathscr{D}_2 =$ `Fasttrack Extended`. The two datasets, summarized in Table 1, were obtained through FastTrack.[4] They were selected for restraining the number of training samples and as representing respectively a low-data setting and a medium-data setting. Both datasets contain daily closing prices of U.S. MFs and ETFs traded on U.S. financial markets, each covering different types of asset classes including stocks, bonds, commodities, currencies and market indexes, or a proxy for a market index. Taken in combination, they cover 19 years of financial market activities and provide an overall view of the whole financial ecosystem. Each TS of these datasets represents the aggregation of multiple individual financial assets. In some of these TS, like `VFICX`, the aggregation of these individual TS is subject to vary over time with respect to management activities associated with these funds.

$$MASE(\tilde{X}, X) = \frac{1}{H}\sum_{i=1}^{H}\frac{|x_{T+i} - \tilde{x}_{T+i}|}{\frac{1}{T+H-s}\sum_{j=s+1}^{T+H}|x_j - x_{j-m}|} \tag{18}$$

$$THEILU(\tilde{X}, X) = \frac{1}{H}\sum_{i=1}^{H}\frac{\sqrt{|x_{T+i} - \tilde{x}_{T+i}|^2}}{\frac{1}{T+H-s}\sum_{j=s+1}^{T+H}\sqrt{|x_{T+i} - \tilde{x}_{T+i}|^2}} \tag{19}$$

$$SDILATE(\tilde{X}, X, \tilde{X}') = \frac{(\alpha)loss_{shape}(\tilde{X}, X) + (1-\alpha)loss_{time}(\tilde{X}, X)}{(\alpha)loss_{shape}(\tilde{X}', X) + (1-\alpha)loss_{time}(\tilde{X}', X)} \tag{20}$$

$$MDA(\tilde{X}, X) = \frac{1}{N}\sum_{i=0}^{\tau} sign(\tilde{X}_{t:t+i} - X_{t-1}) = sign(X_{t:t+i} - X_{t-1}) \tag{21}$$

To train each model, we carried out an evaluation on a rolling-forecasting-origin cross-validation, setting the number of time steps $\tau$ to 21 days for simulating forecasting on a monthly basis. All models were trained on normalized TS using the

[4] https://investorsfasttrack.com.

**Table 2**
Average forecasting performance of tested models on the `Fasttrack` dataset.

| *Model* | FAST TRACK | | | |
|---|---|---|---|---|
| | *MASE* | *THEILU* | *sDILATE* | *MDA* |
| Naive | $1.0000 \pm 0.0000$ | $1.0000 \pm 0.0000$ | $1.0000 \pm 0.0000$ | $0.0180 \pm 0.0149^{****}$ |
| AR | $1.0707 \pm 0.1517^{****}$ | $1.0757 \pm 0.1577^{****}$ | $1.1819 \pm 0.3560^{****}$ | $0.5085 \pm 0.1469^{***}$ |
| ARIMA | $1.0030 \pm 0.1205^{t}$ | $1.0133 \pm 0.1204^{t}$ | $1.0412 \pm 0.2457^{*}$ | $\mathbf{0.5817 \pm 0.1834}$ |
| LSTM | $1.3399 \pm 0.6020^{****}$ | $1.3405 \pm 0.6332^{****}$ | $2.1941 \pm 2.5503^{****}$ | $0.4861 \pm 0.1624^{****}$ |
| LSTM-A | $1.5708 \pm 0.6607^{****}$ | $1.5088 \pm 0.6261^{****}$ | $2.6648 \pm 2.4887^{****}$ | $0.4355 \pm 0.1642^{****}$ |
| WaveNet | $1.5936 \pm 0.7655^{****}$ | $1.6093 \pm 0.8133^{****}$ | $3.2449 \pm 3.7111^{****}$ | $0.4844 \pm 0.1606^{****}$ |
| 1-BEATS | $2.9653 \pm 1.3327^{****}$ | $2.8485 \pm 1.3271^{****}$ | $9.8578 \pm 9.2178^{****}$ | $0.4307 \pm 0.1490^{****}$ |
| STNN | $\mathbf{0.9852 \pm 0.0693}$ | $\mathbf{0.9920 \pm 0.0756}$ | $\mathbf{0.9897 \pm 0.1484}$ | $\underline{\mathbf{0.5942}} \pm \underline{\mathbf{0.1816}}$ |
| STNN-R | $\mathbf{0.9860 \pm 0.0785}$ | $0.9900 \pm 0.0791^{*}$ | $0.9863 \pm 0.1431^{*}$ | $0.5450 \pm 0.1965^{*}$ |
| STNN-D | $1.0812 \pm 0.2957^{***}$ | $1.0808 \pm 0.2765^{**}$ | $1.2439 \pm 0.8131^{**}$ | $0.5585 \pm 0.1533^{t}$ |
| STANN | $\underline{\mathbf{0.9792}} \pm \underline{\mathbf{0.1045}}$ | $\underline{\mathbf{0.9828}} \pm \underline{\mathbf{0.1114}}$ | $\mathbf{0.9783 \pm 0.2174}$ | $\mathbf{0.5363 \pm 0.1914}$ |
| STANN-R | $\mathbf{0.9806 \pm 0.0784}$ | $\mathbf{0.9863 \pm 0.0804}$ | $\mathbf{0.9793 \pm 0.1562}$ | $\mathbf{0.5864 \pm 0.1873}$ |
| STANN-D | $\mathbf{0.9864 \pm 0.0381}$ | $\mathbf{0.9870 \pm 0.0374}$ | $\underline{\mathbf{0.9756}} \pm \underline{\mathbf{0.0707}}$ | $0.5642 \pm 0.1956^{t}$ |

Averaged forecasting results of the 21-day multivariate trajectory forecasts for both datasets. Boldface indicates the best methods who was determined by the Wilcoxon signed-rank test with significance level of $p$-value $< 0.10$. We also indicate the statistical significance of the difference from the best-performing model on the associated metric (t: $p \leq 0.10$; *: $p \leq 0.05$; **: $p \leq 0.01$; ***: $p \leq 0.001$; ****: $p \leq 0.0001$). Underlining is used to indicate the best-performing model, comparing the significance level on all metrics.

interquartile range method. Produced forecasts were unscaled back to the original TS scales to measure the forecast's error. All DNN-based models were trained using stochastic gradient descent (SGD) with Adam [35] and a learning rate scheduler [36]. The number of epochs, learning rate and other model hyperparameters, such as the optimal training window or the number of hidden layers and the number of hidden neurons for DNN-based models, were determined by a Bayesian hyperparameter search [37].

### 4.2. Time series models

For fairness of comparison, we considered only models that can forecast multivariate TS directly, with the exception of two baseline models. The models used are as follows:

1. **Naive:** A simple heuristic that assumes the $\tau$ future steps will be the same as the last previously observed.
2. **AR:** A classical univariate autoregressive process in which each TS is forecasted individually. The prediction is a linear function of past $l$ lags.
3. **ARIMA:** An autoregressive integrated moving average model that forecasts each TS individually. Implementation of ARIMA was done with [38] to automatize the selection of the best parameterization over the training set.
4. **LSTM:** A long short-term memory model that forecasts $\tau$ steps ahead in an iterative fashion [29]. LSTM with hidden layers and the number of hidden neurons were considered.
5. **LSTM-A:** The same model as LSTM but with an added softmax attention layer to weight the importance of each past latent state for forecasting the next step-ahead.
6. **WaveNet:** A convolutional neural network using causal convolutions [39].
7. **1-BEATS:** A member of the neural basis expansion analysis for time series forecasting ensemble model presented in [24].
8. **STNN:** The closest model to ours. STNN can be considered as a particular case of our model, i.e., our model with $k = 1$. The two extensions of STNN (STNN-R and STNN-D) [34] were also considered. The Pearson correlations between TS were computed over the training set to define $W$. We used the same training strategy as for STANN, i.e., modeling the variation only and training the model end-to-end to establish a fair comparison between model architectures.
9. **STANN:** The model proposed in this paper. The two extensions presented in Section 3.4 were also considered. The extensions expressed in Eq. (15) and Eq. (16) are denoted by STANN-R and STANN-D, respectively. Pearson correlation was used to define $W$.

### 4.3. Forecasting performance

Our experimental results are summarized in Table 2 and Table 3. First, we analyze the average performance of all the models and the statistical significance of the results obtained. Our model outperforms the DNN-based and statistical baselines in terms of all metrics on both datasets. The values of these metrics also indicate the superiority of the training framework of STNN and STANN as compared to the other models evaluated. By using the proposed attention mechanism and the TS decomposition approach of N-BEATS, STANN improves on the performance of its base model (STNN).

**Table 3**
Average forecasting performance of tested models on the `Fasttrack extended` datasets.

| *Model* | FAST TRACK EXTENDED | | | |
|---|---|---|---|---|
| | *MASE* | *THEILU* | *sDILATE* | *MDA* |
| Naive | $1.0000 \pm 0.0000$ | $1.0000 \pm 0.0000$ | $1.0000 \pm 0.0000$ | $0.0128 \pm 0.0082^{****}$ |
| AR | $1.0337 \pm 0.0844^{****}$ | $1.0306 \pm 0.1033^{***}$ | $1.0723 \pm 0.2109^{***}$ | $0.4788 \pm 0.0955^{*}$ |
| ARIMA | $1.0011 \pm 0.0945^{*}$ | $1.0008 \pm 0.1193^{t}$ | $1.0156 \pm 0.2373^{t}$ | $0.2748 \pm 0.1222^{****}$ |
| LSTM | $1.2543 \pm 0.3020^{****}$ | $1.2311 \pm 0.3002^{****}$ | $1.6041 \pm 0.8031^{****}$ | $0.4821 \pm 0.1426^{t}$ |
| LSTM-A | $1.3940 \pm 0.3900^{****}$ | $1.3410 \pm 0.3713^{****}$ | $1.9345 \pm 1.1580^{****}$ | $0.4841 \pm 0.1357^{t}$ |
| WaveNet | $1.3988 \pm 0.5445^{****}$ | $1.4071 \pm 0.5930^{****}$ | $2.3042 \pm 2.4721^{****}$ | $0.4864 \pm 0.1472^{*}$ |
| 1-BEATS | $2.9836 \pm 1.1605^{****}$ | $2.7788 \pm 1.1065^{****}$ | $8.7333 \pm 6.5840^{****}$ | $0.4565 \pm 0.1449^{****}$ |
| STNN | $\mathbf{1.0020 \pm 0.1536}$ | $\mathbf{0.9959 \pm 0.1591}$ | $\mathbf{1.0165 \pm 0.3354}$ | $\mathbf{0.5259 \pm 0.1822}$ |
| STNN-R | $\mathbf{1.0122 \pm 0.1707}$ | $\mathbf{1.0047 \pm 0.1698}$ | $\mathbf{1.0369 \pm 0.3687}$ | $\mathbf{0.5241 \pm 0.1693}$ |
| STNN-D | $\mathbf{0.9814 \pm 0.1147}$ | $\mathbf{0.9791 \pm 0.1255}$ | $\mathbf{0.9743 \pm 0.2495}$ | $\mathbf{0.5401 \pm 0.2052}$ |
| STANN | $\mathbf{0.9832 \pm 0.1023}$ | $\mathbf{0.9814 \pm 0.1084}$ | $\mathbf{0.9750 \pm 0.2148}$ | $\mathbf{0.5360 \pm 0.2030}$ |
| STANN-R | $\mathbf{0.9836 \pm 0.1026}$ | $\mathbf{0.9816 \pm 0.1098}$ | $\mathbf{0.9755 \pm 0.2189}$ | $\mathbf{0.5401 \pm 0.2051}$ |
| STANN-D | $\underline{\mathbf{0.9795}} \pm \underline{\mathbf{0.1016}}$ | $\underline{\mathbf{0.9785}} \pm \underline{\mathbf{0.1096}}$ | $\underline{\mathbf{0.9694}} \pm \underline{\mathbf{0.2176}}$ | $\underline{\mathbf{0.5406}} \pm \underline{\mathbf{0.2055}}$ |

Averaged forecasting results of the 21 days multivariate trajectory forecasts. We highlight the best methods in bold.

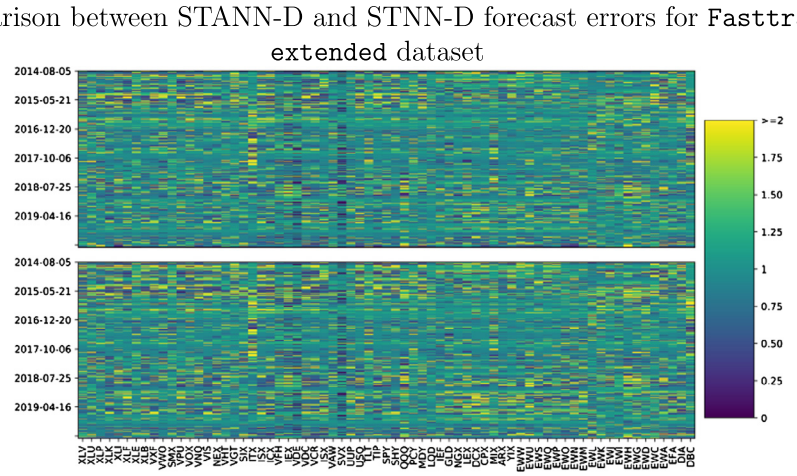


**Fig. 4.** Concatenation of the 21 daily return forecasts of STANN-D (top) and STNN-D (bottom). The absolute scaled error per series is presented.

The augmentation trick used in the STNN and STANN models, presented in Eq. (13) and Eq. (14), is the largest contributing factor behind these results. By exploiting prior knowledge on the relation of these TS, STANN and STNN enhance their ability to forecast TS by "virtually" increasing the number of training samples despite using a shared latent state like LSTM and Wavenet. These results are very promising, considering that (1) our approach achieved such results using a relatively small number of TS; (2) it was trained solely using historical prices. It is not surprising that DNN-based models (WAVENET, LSTM, LSTM-A, 1-BEATS) underperform compared to statistical baselines when trained in this setting, given their large parameterization and the small number of training samples at their disposal. Our results show that the augmentation trick of STNN and STANN appears to be a solution to the lack of training samples when such models are trained in a multivariate setting. We point out that, contrary to [18], we did not achieve similar MASE for the one-step-ahead forecast. We observe that during model training, we achieved similar results but the accuracy quickly dropped after the first 5 steps ahead in the first few epochs to yield a better overall forecast on the whole trajectory. Hence, there appears to be a trade-off between short-term forecast accuracy and the longer-term forecast accuracy when optimizing DNN-based models.

We can qualitatively compare our models by plotting the absolute scaled error of the individual point forecast (IPF), i.e., $\frac{|x_{T+i} - \tilde{x}_{T+i}|}{\frac{1}{T+H-m}\sum_{j=m+1}^{T+H} |x_j - x_{j-m}|}$, for all the TS forecast, and comparing where our model fails. We observe that our approach is relatively consistent at forecasting the trajectory of each asset (Fig. 4), but the majority of the residuals appear to occur in an epistemic fashion, i.e., the TS forecasting difficulty varies over time. Our proposed attention mechanism slightly increases the forecast accuracy in these episodes of forecast instability over its base model, which explains the majority of the additional average gain in accuracy.

By looking at the ability of the evaluated models to predict the trend of the out-of-sample trajectory, we observe in Fig. 5 that the STANN- and STNN-based models outperform baseline models but the STANN-based models show less variance in their results when compared to the different extensions of the STNN-based models. Given that these models tend to forecast the appropriate trend more accurately, this explains why our proposed framework outperforms other baseline models.

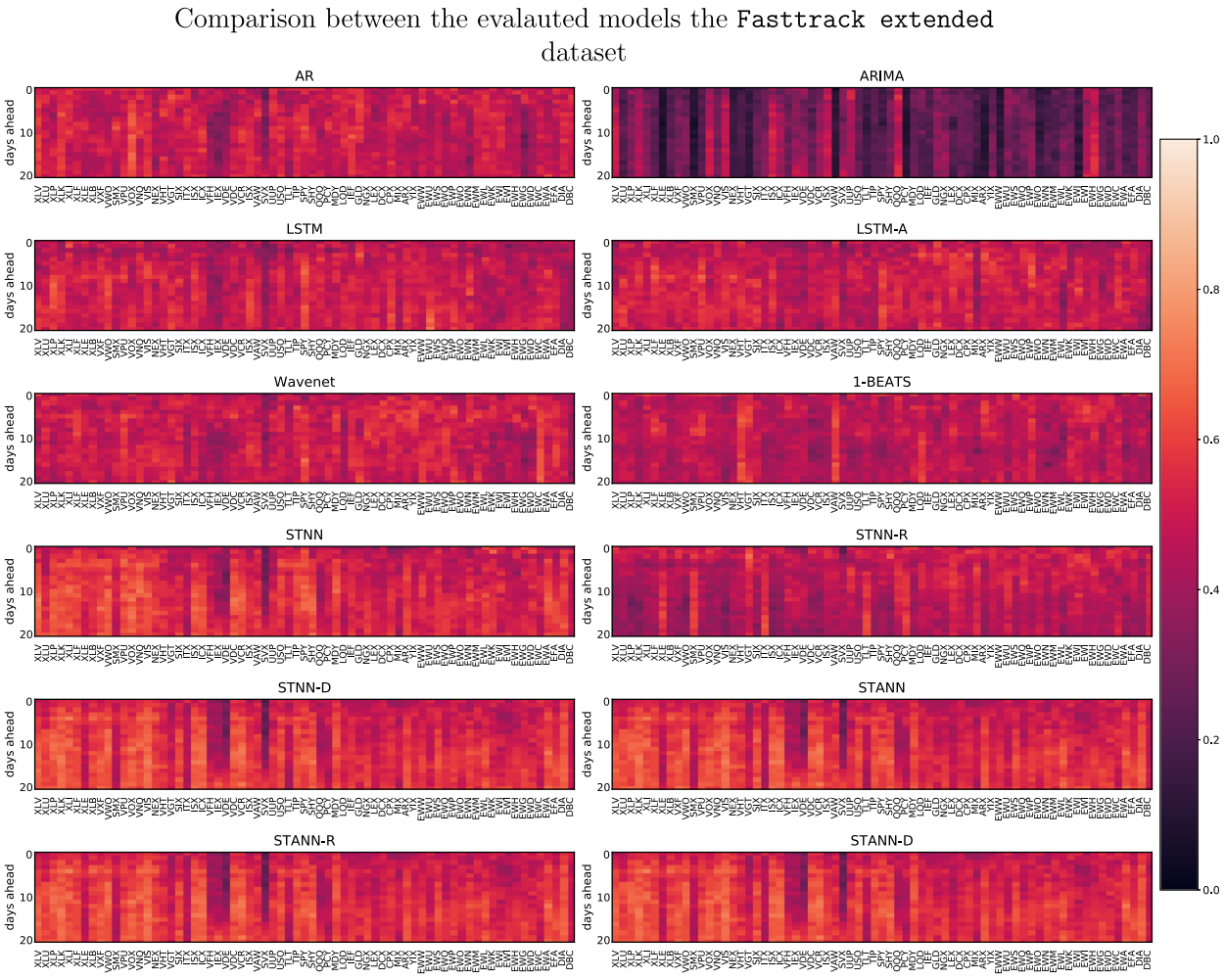


**Fig. 5.** Directional accuracy for each $t$ steps ahead of the evaluated models. The darker the region, the less precise the model is at forecasting the trend of this particular asset.

In Fig. 6, we illustrate, showing the autoregressive order of a sample of the STANN-R model, that our model generates a dynamic process thanks to its attention mechanism. We can see that the model oscillates between AR(1) and AR(2) processes, although it remains more often at an AR(1) process. Interestingly, we can identify time spans for certain assets that are dominated by an AR(2) process and other regions dominated by an AR(1) process. This phenomenon is similar to what regime switching (RS) models [40] enforce as prior when modeling the TS. Our approach differs in that we do not have to specify the number of regimes, nor the AR order *a priori*. However, we notice that the choice of hyperparameters and model architecture can lead to vastly different results, with instances converging to either stationary or higher-AR-order solutions. Hence, further studies are needed to determine how this attention mechanism permits modeling of regime switches within its latent states.

Finally, we also performed an ablation study to show the effect that the TS decomposition technique and the attention mechanism have on the model. To this end, we plotted the values of MASE metrics of each step-ahead forecast of model STNN-D and STANN-D with one of the two components removed. The plot, presented in Fig. 7, shows the reference model (top left), our proposed model (top right), our model without ACTM (bottom left) and our model without the N-BEATS architecture (bottom right). The plot shows that using the attention mechanism alone without TS decomposition increases the overall forecast error but reduces the error propagation often found in recursive approaches. When combined with the TS decomposition architecture presented in [24], we observe a significant error reduction for the last 14 days of the forecast trajectory. Regardless of the number of steps ahead forecast, our approach is significantly better than its base model at reducing IPF ****.

### *4.4. Added value for autonomous decision making*

We conducted some preliminary studies to demonstrate the utility of the proposed model to help autonomous decision making. Specifically, we show how the improved forecasting accuracy achieved by our model enhances the performance

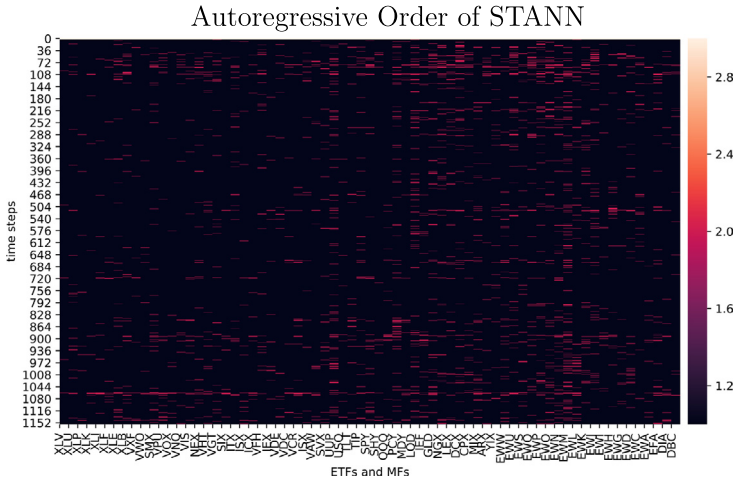


**Fig. 6.** In-sample autoregressive order of a random instance of the STNN-R models taken from the `Fasttrack extended` evaluation. The left axis corresponds to past time steps used to train the model, with $0^{\text{iem}}$ steps being the furthest away from the prediction date; the bottom axis shows the securities forecast. Brighter colors indicate a higher autoregressive order for the time step in question.

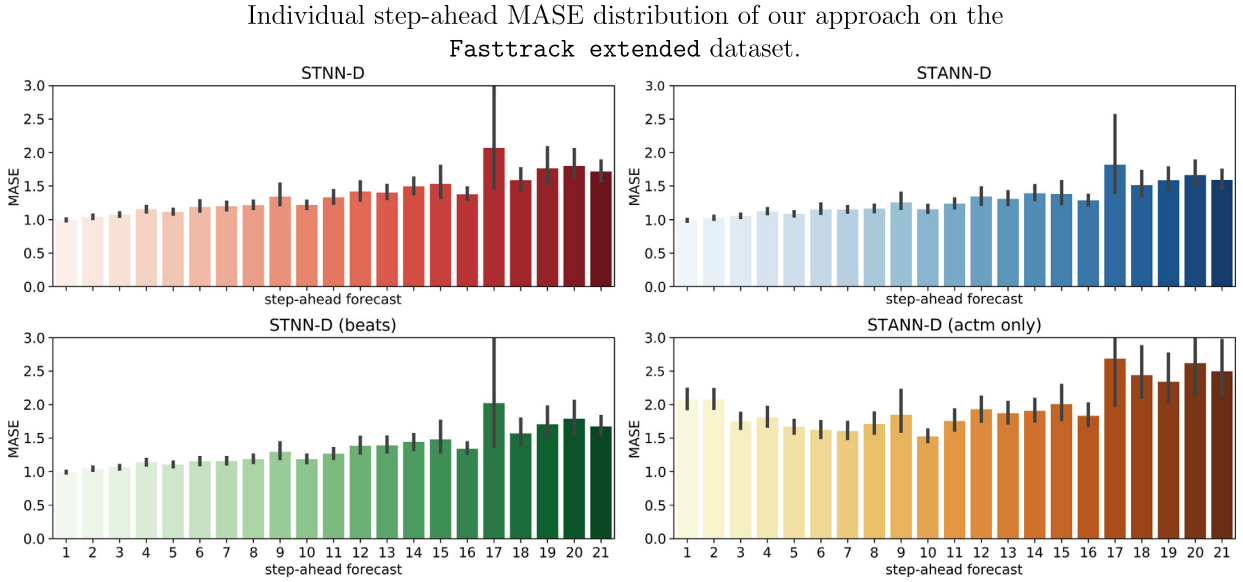


**Fig. 7.** An ablation study of STANN-D is presented. The median absolute scaled error for STANN-D is equal to 0.9671 and for STNN-D, to 0.9810.

of autonomous trading strategies. The challenges of this question are threefold: (1) TS models are not decision systems in themselves. (2) We must rely on a trading strategy that considers forecast as a proxy to compare forecasting models. Given the myriad of strategies that exist, selecting one trading strategy over another is highly subjective and may lead to ambiguous results [41]. (3) Each trading strategy has its own sensitivity and some trading strategies bypass the forecasting step by directly generating trade positions [42]. This last point is particularly important, as different trading strategies will produce different allocations depending on the same input data. Consequently, the sensitivity of trading strategies, and the state space on which they operate to establish trade signal plays a significant role in determining the excess return. This results in multiple issues that render it difficult to isolate the exact role that forecast accuracy has on various trading approaches, especially more sophisticated ones. It is necessary to point out that an extensive study of these issues is beyond the scope of this paper. However, we discuss briefly these issues and point out similarities and differences that our approach has over existing ones.

One of the most popular frameworks for portfolio optimization, i.e., the traditional mean/variance framework [43], is a typical framework where the expected returns and variances are estimated and asset allocation decided based on an optimization procedure that aims to minimize a loss function. This approach is known to produce mixed results when exposed to noisy forecasts [44]. Alternatively a simple strategy that generates buy and trade signals like in [45] can be made to translate the trajectory predicted in a trading action. The advantage of this approach is that they are model-

**Table 4**
Portfolio performance metrics of the three types of strategy, evaluated on the same time horizon utilized in Table 1. The best strategy is highlighted in bold.

Performance of various portfolio strategies built on the evaluated models

| *Model* | FAST TRACK | | | | FAST TRACK EXTENDED | | | |
|---|---|---|---|---|---|---|---|---|
| | *Sharpe* | *Max drawdown* | *Mean return* | *Profit* | *Sharpe* | *Max drawdown* | *Mean return* | *Profit* |
| **BL strategy** | | | | | | | | |
| AR | 0.37 | −2.41% | 0.34% | 39.87% | 0.80 | −2.30% | 0.24% | 14.82% |
| ARIMA | 0.47 | −2.40% | 0.35% | 41.74% | 0.97 | −2.47% | 0.30% | 18.79% |
| LSTM | 0.67 | −1.87% | 0.39% | 46.71% | 0.82 | −2.84% | 0.28% | 17.27% |
| LSTM-A | 0.49 | −2.23% | 0.36% | 42.62% | 1.13 | −3.02% | 0.35% | 21.91% |
| WaveNet | −0.02 | −48.52% | 0.32% | 30.58% | 0.52 | −6.90% | 0.27% | 16.43% |
| 1-BEATS | −0.04 | −39.76% | 0.30% | 29.46% | 0.05 | −10.02% | 0.10% | 5.19% |
| STNN | 0.49 | −2.21% | 0.35% | 41.95% | 0.76 | −2.61% | 0.22% | 13.51% |
| STNN-R | 0.50 | −2.21% | 0.35% | 42.16% | 1.11 | −4.13% | 0.54% | 35.70% |
| STNN-D | 0.48 | −2.13% | 0.35% | 41.72% | 0.74 | −2.64% | 0.22% | 13.52% |
| STANN | 0.48 | −2.19% | 0.35% | 41.70% | 0.73 | −2.57% | 0.22% | 13.26% |
| STANN-R | 0.47 | −2.20% | 0.35% | 41.67% | 0.73 | −2.67% | 0.22% | 13.37% |
| STANN-D | 0.49 | −2.23% | 0.35% | 41.92% | 0.72 | −2.66% | 0.22% | 13.22% |
| **Optimal** | **3.97** | **-1.92%** | **2.79%** | **1436.73%** | **2.89** | **-5.30%** | **2.27%** | **263.39%** |
| **Simple strategy** | | | | | | | | |
| AR | −0.26 | −20.70% | 0.18% | 16.60% | 0.38 | −16.56% | 0.37% | 21.59% |
| ARIMA | 0.47 | −11.58% | 0.50% | 62.22% | 0.51 | −11.85% | 0.40% | 24.63% |
| LSTM | 0.15 | −29.69% | 0.44% | 47.40% | **0.76** | −21.08% | **0.58%** | **37.39%** |
| LSTM-A | −0.06 | −35.53% | 0.28% | 27.54% | **0.53** | −17.18% | **0.52%** | **31.51%** |
| WaveNet | −0.02 | −48.52% | 0.32% | 30.58% | 0.41 | −15.28% | 0.35% | 20.35% |
| 1-BEATS | −0.04 | −39.76% | 0.30% | 29.46% | 0.21 | −17.41% | 0.31% | 16.18% |
| **STNN** | 0.49 | −11.47% | 0.51% | 63.55% | 0.40 | −15.64% | 0.37% | 21.52% |
| STNN-R | **0.54** | −12.26% | 0.50% | 62.24% | **0.70** | **-0.38%** | 0.18% | 10.95% |
| **STNN-D** | **0.75** | **-6.43%** | **0.61%** | **81.36%** | 0.48 | **-10.73%** | 0.41% | 24.70% |
| **STANN** | **0.52** | −19.59% | **0.56%** | **70.75%** | 0.50 | **-10.86%** | **0.43%** | **25.82%** |
| STANN-R | 0.45 | −12.07% | 0.49% | 60.64% | 0.48 | **-10.73%** | 0.41% | 24.70% |
| **STANN-D** | **0.66** | **-8.99%** | **0.55**% | **71.13%** | 0.48 | **-10.73%** | 0.41% | 24.64% |
| Equal weight | 0.47 | −11.58% | 0.50% | 62.22% | 0.48 | −10.73% | 0.41% | 24.70% |

agnostic, i.e., they do not depend on the model choice to apply the prediction and there exists correlation between having good estimates of the returns and the performance of a trading strategy. However, past profits can vanish very rapidly if the trading strategy makes a bad allocation at a bad time. Even if a TS model's accuracy is good on average, it suffices for the TS forecast to be *bad* at the wrong moment to lose profits from past months. Hence, the timing of the forecasts plays a significant role in the performance of these strategies.

In comparison, other DNN-based strategies can be used to compare our TS model. These approaches generate directly trade signals within their architecture. This is either done in typical supervised setting like in [46,47] where a model aims to solve classification task with each class corresponding to a trade action. In this setting, a trade action must be labeled for each input data. Alternatively, RL approaches like [48,49,42] can be applied where a model generates trade actions based on their perceived state of their environment. These methods learn a policy which generates a probability distribution over possible trade actions based on a state space and are trained to maximize a reward function. This state space can be fabricated by a selection of different features as in [42]. In this setting, we do not require labels for each input data, but we must specify a reward function for assessing the quality of the action chosen based on its environment. To compare the effect of TS models accuracy using either of these two approaches, TS forecast must be included within their input data.

However, additional issues arise from the use of these methods. As stated in [42], RL approaches can exhibit "slow learning and need a lot of samples to obtain an optimal policy." Similarly, this can also be the case of a DNN-based model depending on the model architecture. Unless we have at our disposal, the historical prediction for each time interval at all time points, using DNN and RL approaches is impractical as the amount of training samples to train these methods is diminished by the pace at which we can produce forecasts. Hence, for all DNN-based methods (LSTM, LSTM-A, WAVENET, STNN and STANN), one must resort to zero-shot forecasting, i.e., without explicit retraining on that target data, to produce these forecasts in a reasonable amount of time to justify the use of RL approaches. This, however, augments exposure to concepts drift effect that occurs when not retraining model [11], which, in turns, tends to lower the average accuracy of the TS model [50]. Hence, two additional factors can influence the performance of this approach: (1) sensibility to hyperparameters and model architecture of the trading strategy, (2) the number of training samples at our disposal where a forecast has been made. To prevent the leakage of these additional issues, we restrict our experimental analysis to two traditional approaches, which we describe below.

In our experiments, we studied and observed relations between the accuracy of the TS forecast and the excess return of a strategy. Results for various financial metrics of two common strategies are presented in Table 4 on both datasets. Here, we

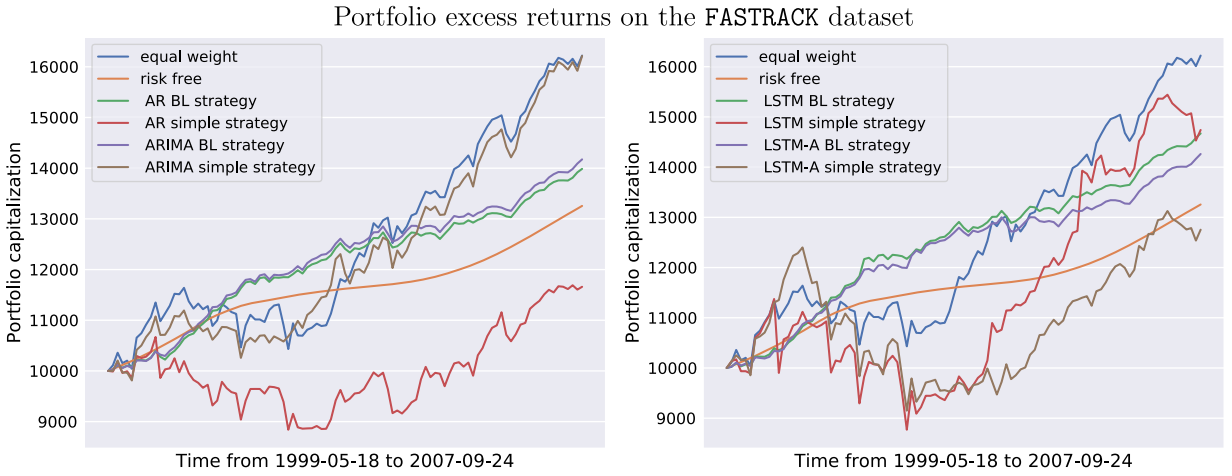


**Fig. 8.** Portfolio performances based upon alternative statistical baselines (left) and RNN-based models (right).

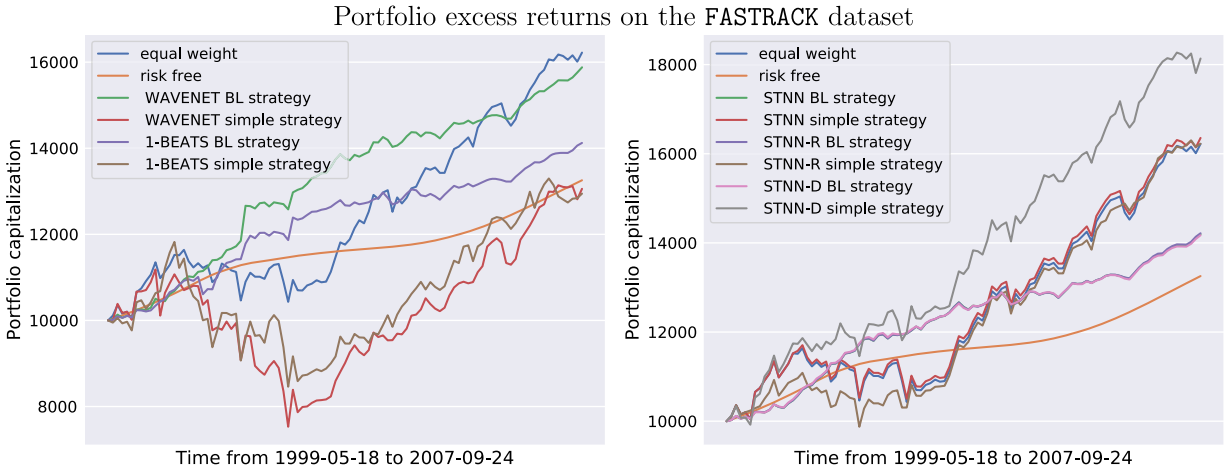


**Fig. 9.** Portfolio performances based on alternative DNN models (left) and STNN-based models (right).

include the annualized Sharpe ratio $sharpe = \frac{E[r_{portfolio} - r_{riskfree}])}{\sigma_{portfolio}} \times \sqrt{\mathscr{S}}$, where $r_{portfolio}$ is the return of the portfolio, $r_{riskfree}$ is the return of a risk-free strategy, $\sigma_{portfolio}$ is the standard deviation of the excess return of the portfolio and $\mathscr{S} = \tau/252$ is an annualization factor, with 252 being the average number of active market days per year. We also include the maximum drawdown, which is the maximum loss observed from a peak to a trough of a portfolio; the mean return per trade and the total profit after the observation periods. We also present the excess return graph of different portfolio strategies built on the evaluated models for the FASTTRACK dataset in Figs. 8, 9 and 10.

We used two proxy trading strategies to carry out our evaluation.

(1) *Black-Litterman (BL) strategy*: The first is a simple, efficient frontier optimization trading strategy [43] based on the BL allocation model [51], in which we maximized the Sharpe ratio. The expected returns are determined by the TS forecasts and implied market returns and the estimated risk was computed respectively by the in-sample returns and covariance.[5]

(2) *Simple:* a simple trading strategy where one invests equally in each of the securities that the TS model predicted would increase in price. If the trend signal is negative and assets were allocated, we consider the forecast as a *"sell"* signal. A positive forecast trend is interpreted as a *"buy"* signal. The allocation weights were normalized according to the strength of the trend signal using a softmax function.

[5] A long-only portfolio constraint was added to ensure that the allocation weights remain between 0 and 1. In some instances this constraint is too restrictive and the convex optimization used in this approach fails. In such cases, we permit the optimization to consider the short position; i.e., allocation weights remain between -1 and 1, and do not consider the short position in our allocation.

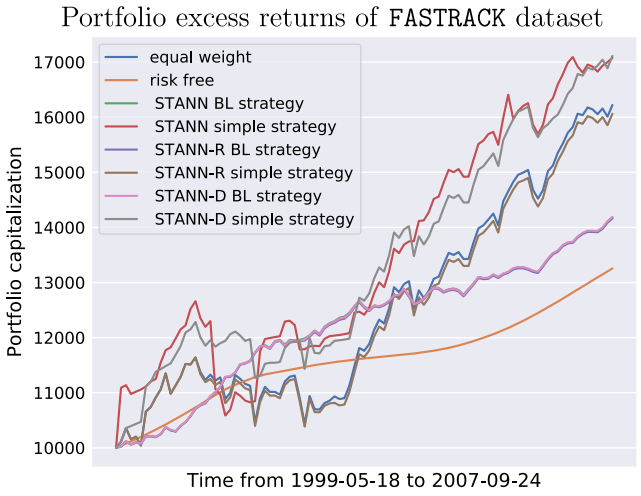


**Fig. 10.** Portfolio performance based upon STANN-based models.

We compare these strategies with the same "optimal" BL strategy where the expected returns and covariance matrix are known in advance to simulate how a perfect TS model would have performed for this strategy. We also include an equal-weight portfolio where we invest in each asset equally.

Allocations of assets were reconsidered on a 21-day basis to simulate how an autonomous trading strategy would change its portfolio allocation over time depending on the financial context. We used a $10,000 initial portfolio value and bought whole security shares, with the leftover money being at a risk-free rate determined by the 30-day U.S. Treasury bill rate. For the `Fasttrack` dataset, we used the U.S. 3-Month Treasury rate rather than the 1-month rate for the risk-free strategy, given that no data for the U.S. 1-Month Treasury rate before 2001-07-31 is available from the public and proprietary data sources at our disposal. Transaction fees were assumed to be null and dividends were accounted for within the adjusted closing price.

It can be seen that perfect predictions yield extremely significant returns for BL strategies on both datasets. However, although LSTM and LSTM-A are poor forecasters with respect to all metrics, the strategy built on them is among the top performers. Identifying the principal cause of this phenomenon is not a trivial matter. However, efficient frontier optimization methods are known to produce mixed results when the forecasts are too noisy [44]. At this level of accuracy, none of the TS models evaluated are sufficiently robust by themselves to be considered in a BL strategy. However, we can observe that there are significant gains to be made if one could identify the condition where a TS forecaster helps a trading strategy, as each strategy overperforms the baselines at certain moments.

The performance of the simple strategy shows the importance of having a more accurate TS model for autonomous trading strategies. TS models that forecast more accurately, like the STNN and STANN-based approaches, are among the top performers on both datasets with respect to most metrics, especially when few TS are considered. When more TS are considered (`Fasttrack extended`), the simple strategy is too naive to select a meaningful subset of securities and will yield performance similar to an equal-weight portfolio. However, since STNN and STANN-based models forecast the trend more accurately, the maximum drawdowns of the simple strategy based on these models were much smaller compared to the LSTM and LSTM-A strategies, which made allocations over a smaller set of securities at a time. Hence, a simple naive strategy can perform relatively well within a curated set of TS using a better forecaster, but will not scale effectively when the number of assets considered is too large.

## 5. Discussion

This paper proposed a new self-supervised deep generative model (STANN) for forecasting multivariate TS conjointly, which explicitly models the interactions between TS. We introduced a novel attention-based mechanism that enhances the capability of any RNN based on the DFG framework. We showed how this attention-based mechanism increases the set of probability distributions that can be modeled by permitting modeling of non-stationary distributions. Incorporating this into our model, we presented one general approach and two extensions for considering interrelations between TS. We showed that when these interrelations are incorporated, we can fit these DNN-based models even where little training data exists. Experiments were performed on two financial datasets covering more than 19 years of market history. Our experiments indicate that STANN provides a more effective learning framework than either DNN-based approaches or statistical baselines. We showed that this class of models perform wells in both low- and medium-data settings and that our proposed attention mechanism helps improve forecasting performances over its base model. Finally we illustrated how the use of a forecaster improves autonomous trading strategies.

We would like to emphasize the limited understanding of the relation between our model effectiveness and the selection of HPs. Indeed, a mis-selection of HPs can have a great impact on a model's performance, potentially hindering its application at a large scale. Hence, we advocate the pursuit of future work to enlarge our theoretical understanding of this class of models, as well as testing to determine whether similar results can be achieved at larger scales and for other TS settings.

## CRediT authorship contribution statement

**Philippe Chatigny:** Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Resources, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Jean-Marc Patenaude:** Data curation, Funding acquisition, Resources, Supervision, Writing – review & editing. **Shengrui Wang:** Funding acquisition, Project administration, Resources, Supervision, Writing – review & editing.

## Declaration of competing interest

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

## Acknowledgements

This work was supported in part by NSERC CRD - Quebec Prompt - Laplace Insights - EAM - joint program under the grants CRDPJ 537461-18 and 114-IA-Wang-DRC 2019 to Dr. S. Wang, by Mitacs Acceleration under Grant IT13429 to Dr. S. Wang and P. Chatigny and by FQRNT under Grant B2 270188 to P. Chatigny.

## References

[1] S. Makridakis, R.J. Hyndman, F. Petropoulos, Forecasting in social settings: the state of the art, Int. J. Forecast. 36 (2020) 15–28.
[2] M. Piotr, Y. LeCun, Dynamic factor graphs for time series modeling, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2009, pp. 128–143.
[3] P. Mirowski, Time series modeling with hidden variables and gradient-based algorithms, Ph.D. Dissertation, Department of Computer Science, Courant Institute of Mathematical Sciences, New York University, 2011.
[4] D. Koller, N. Friedman, Probabilistic Graphical Models: Principles and Techniques, MIT Press, 2009.
[5] Y. Bengio, Learning deep architectures for AI, Found. Trends Mach. Learn. 2 (2009) 1–127.
[6] R.J. Hyndman, G. Athanasopoulos, Forecasting: Principles and Practice, OTexts, 2018.
[7] H.-F. Yang, T.S. Dillon, Y.-P.P. Chen, Optimized structure of the traffic flow forecasting model with a deep learning approach, IEEE Trans. Neural Netw. Learn. Syst. 28 (2016) 2371–2381.
[8] M.D. Olagoke, A. Ayeni, M.A. Hambali, Short term electric load forecasting using neural network and genetic algorithm, Int. J. Appl. Inf. Syst. 10 (2016) 22–28.
[9] S. Makridakis, E. Spiliotis, V. Assimakopoulos, Statistical and machine learning forecasting methods: concerns and ways forward, PLoS ONE 13 (2018) e0194889.
[10] O.B. Sezer, M.U. Gudelek, A.M. Ozbayoglu, Financial time series forecasting with deep learning: a systematic literature review: 2005–2019, Appl. Soft Comput. 90 (2020) 106181.
[11] I. Žliobaitė, Learning under concept drift: an overview, preprint, arXiv:1010.4784, 2010.
[12] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, G. Cottrell, A dual-stage attention-based recurrent neural network for time series prediction, in: International Joint Conference on Artificial Intelligence, 2017.
[13] M. Bruche, C. González-Aguado, Recovery rates, default probabilities, and the credit cycle, J. Bank. Finance 34 (2010) 754–764.
[14] A. Marshall, Principles of Economics: Unabridged, eighth edition, Cosimo, Inc., 2009.
[15] Y. Bengio, P. Simard, P. Frasconi, et al., Learning long-term dependencies with gradient descent is difficult, IEEE Trans. Neural Netw. 5 (1994) 157–166.
[16] Y. Rubanova, T.Q. Chen, D.K. Duvenaud, Latent ordinary differential equations for irregularly-sampled time series, in: Advances in Neural Information Processing Systems, 2019, pp. 5321–5331.
[17] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using RNN encoder–decoder for statistical machine translation, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP, 2014, pp. 1724–1734.
[18] A. Borovykh, S. Bohte, C.W. Oosterlee, Dilated convolutional neural networks for time series forecasting (October 25, 2018), J. Comput. Financ., forthcoming. Available at SSRN: https://ssrn.com/abstract=3272962.
[19] C. Chatfield, et al., Apples, oranges and mean square error, Int. J. Forecast. 4 (1988) 515–518.
[20] R.J. Hyndman, A.B. Koehler, Another look at measures of forecast accuracy, Int. J. Forecast. 22 (2006) 679–688.
[21] S.S. Rangapuram, M.W. Seeger, J. Gasthaus, L. Stella, Y. Wang, T. Januschowski, Deep state space models for time series forecasting, in: Advances in NIPS, 2018, pp. 7785–7794.
[22] S. Smyl, A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting, Int. J. Forecast. 36 (2020) 75–85, https://doi.org/10.1016/j.ijforecast.2019.03.017, https://linkinghub.elsevier.com/retrieve/pii/S0169207019301153.
[23] S. Makridakis, E. Spiliotis, V. Assimakopoulos, The M4 competition: results, findings, conclusion and way forward, Int. J. Forecast. 34 (2018) 802–808.
[24] B.N. Oreshkin, D. Carpov, N. Chapados, Y. Bengio, N-beats: neural basis expansion analysis for interpretable time series forecasting, in: International Conference on Learning Representations, 2019.
[25] L.B. Godfrey, M.S. Gashler, Neural decomposition of time-series data for effective generalization, IEEE Trans. Neural Netw. Learn. Syst. 29 (2017) 2973–2985.
[26] J. Hansen, R. Nelson, Forecasting and recombining time-series components by using neural networks, J. Oper. Res. Soc. 54 (2003) 307–317.
[27] J.P. Keener, The Perron–Frobenius theorem and the ranking of football teams, SIAM Rev. 35 (1993) 80–93.
[28] P.W. Glynn, P.Y. Desai, A probabilistic proof of the Perron-Frobenius theorem, preprint, arXiv:1808.04964, 2018.
[29] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (1997) 1735–1780.

[30] A. Graves, Adaptive computation time for recurrent neural networks, preprint, arXiv:1603.08983, 2016.
[31] C. Olah, S. Carter, Attention and augmented recurrent neural networks, Distill (2016), https://doi.org/10.23915/distill.00001, http://distill.pub/2016/augmented-rnns.
[32] G. Tesauro, Temporal difference learning and TD-Gammon, Commun. ACM 38 (1995) 58–68.
[33] A. Schwendener, The estimation of financial markets by means of a regime-switching model, Ph.D. thesis, University of St. Gallen, 2010.
[34] A. Ziat, E. Delasalles, L. Denoyer, P. Gallinari, Spatio-temporal neural networks for space-time series forecasting and relations discovery, in: 2017 IEEE ICDM, IEEE, 2017, pp. 705–714.
[35] D.P. Kingma, J. Ba Adam, A method for stochastic optimization, in: Y. Bengio, Y. LeCun (Eds.), 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015, http://arxiv.org/abs/1412.6980, 2015.
[36] I. Loshchilov, F. Hutter, SGDR: stochastic gradient descent with warm restarts, in: International Conference on Learning Representations (ICLR) 2017 Conference Track, 2017.
[37] M. Balandat, B. Karrer, D.R. Jiang, S. Daulton, B. Letham, A.G. Wilson, E. Bakshy, BoTorch: programmable ayesian optimization in PyTorch, preprint, arXiv:1910.06403, 2019.
[38] T.G. Smith, et al., pmdarima: ARIMA estimators for Python, http://www.alkaline-ml.com/pmdarima, 2017, Online.
[39] A. Van Den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A.W. Senior, K. Kavukcuoglu, Wavenet: a generative model for raw audio, SSW 125 (2016).
[40] J.D. Hamilton, A new approach to the economic analysis of nonstationary time series and the business cycle, Econometrica, J. Econ. Soc. (1989) 357–384.
[41] R. Roll, Ambiguity when performance is measured by the securities market line, J. Finance 33 (1978) 1051–1069.
[42] Z. Zhang, S. Zohren, S. Roberts, Deep reinforcement learning for trading, J. Financ. Data Sci. 2 (2) (2020) 25–40.
[43] M. Rubinstein, Markowitz's "portfolio selection": a fifty-year retrospective, J. Finance 57 (2002) 1041–1045.
[44] G. Connor, Sensible return forecasting for portfolio management, Financ. Anal. J. 53 (1997) 44–51.
[45] W. Bao, J. Yue, Y. Rao, A deep learning framework for financial time series using stacked autoencoders and long-short term memory, PLoS ONE 12 (2017) e0180944.
[46] N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, A. Iosifidis, Deep adaptive input normalization for time series forecasting, IEEE Trans. Neural Netw. Learn. Syst. (2019).
[47] A. Tsantekidis, N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, A. Iosifidis, Forecasting stock prices from the limit order book using convolutional neural networks, in: 2017 IEEE 19th Conference on Business Informatics, vol. 1, (CBI), IEEE, 2017, pp. 7–12.
[48] Z. Zhang, S. Zohren, S. Roberts, DeepLOB: deep convolutional neural networks for limit order books, IEEE Trans. Signal Process. 67 (2019) 3001–3012.
[49] Y. Deng, F. Bao, Y. Kong, Z. Ren, Q. Dai, Deep direct reinforcement learning for financial signal representation and trading, IEEE Trans. Neural Netw. Learn. Syst. 28 (3) (2016) 653–664.
[50] B.N. Oreshkin, D. Carpov, N. Chapados, Y. Bengio, Meta-learning framework with applications to zero-shot time-series forecasting, arXiv preprint, arXiv:2002.02887, 2020.
[51] F. Black, R. Litterman, Asset allocation: combining investor views with market equilibrium, Goldman Sachs Fixed Income Res. 115 (1990).