

## Deep Learning and Econometric Time Series Analysis: An Assessment of Daily Return Forecasts

Theo Berger

Suggested citation:

Berger, Theo. 2025. "Deep Learning and Econometric Time Series Analysis: An Assessment of Daily Return Forecasts." *Journal of Forecasting* 45 (1): 377–90. <https://doi.org/10.25968/opus-3824>.

### Abstract

We provide an in-depth assessment of univariate financial time series analysis via machine learning followed by a thorough discussion beyond the discussion on daily return predictability. We simulate economic time series and present an in-depth assessment of relevant hyperparameter tuning and study the ability of competing deep learning algorithms to capture econometric properties of financial time series. Also, we assess empirical data and discuss competing approaches in comparison with econometric benchmarks, when the data generating process is unknown. As a result, we assess more than 512,000 in-sample and out-of-sample forecasts for different scenarios of competing network architectures. Drawing on realistic sample sizes, we find that recurrent neural networks with one layer describe a solid alternative to econometric autoregressive moving average (ARMA) approach.

Terms of use

CC BY 4.0

This document is made available under these conditions:  
**Creative Commons - CC BY - Namensnennung 4.0 International**  
For more information see:  
<https://creativecommons.org/licenses/by/4.0/deed.de>



## RESEARCH ARTICLE OPEN ACCESS

# Deep Learning and Econometric Time Series Analysis: An Assessment of Daily Return Forecasts

Theo Berger<sup>1,2</sup> 

<sup>1</sup>Faculty of Business and Computer Science, Hochschule Hannover - University of Applied Sciences and Arts, Hannover, Germany | <sup>2</sup>Department of Business and Administration, University of Bremen, Bremen, Germany

**Correspondence:** Theo Berger ([theo.berger@hs-hannover.de](mailto:theo.berger@hs-hannover.de))

**Received:** 4 July 2023 | **Revised:** 3 September 2024 | **Accepted:** 29 September 2025

**Keywords:** deep learning | forecasting | machine learning | risk measurement | time series analysis

## ABSTRACT

We provide an in-depth assessment of univariate financial time series analysis via machine learning followed by a thorough discussion beyond the discussion on daily return predictability. We simulate economic time series and present an in-depth assessment of relevant hyperparameter tuning and study the ability of competing deep learning algorithms to capture econometric properties of financial time series. Also, we assess empirical data and discuss competing approaches in comparison with econometric benchmarks, when the data generating process is unknown. As a result, we assess more than 512,000 in-sample and out-of-sample forecasts for different scenarios of competing network architectures. Drawing on realistic sample sizes, we find that recurrent neural networks with one layer describe a solid alternative to econometric autoregressive moving average (ARMA) approach.

## 1 | Introduction

The steadily growing availability of innovative data sets and the increasing accessibility of computational power triggered the recent episode of Artificial Intelligence related research in the field of economic data. As presented in Kraus et al. (2020), the application of machine learning (ML) to financial data is accompanied by growing interest and a novel string of literature dealing with the application of complex approaches, namely deep learning (DL), to economic data emerged. In this vein, Gu et al. (2020) provide a thorough assessment on competing ML approaches applied to economic panel data and present empirical evidence that ML techniques outperform statistical benchmarks. Avramov et al. (2022) also discuss ML methods applied to cross-sectional data and find that DL presents a fruitful alternative to time series regression. However, when it comes to the assessment of financial time series data, econometric time series analysis provides the underlying information for financial risk measurement and management. Due to the limited amount of historic data, the application of

ML to financial time series describes a staggering task, and a thorough discussion on the application of ML as an alternative to econometric time series analysis describes an existing gap in the literature.

The scope of this paper is to provide an in-depth assessment of ML and DL in the context of univariate time series analysis beyond accurate out-of-sample forecasting performance. We draw on the widely accepted econometric benchmark for univariate time series analysis and provide a stepwise assessment of DL techniques to adequately model the conditional mean of financial time series. Along the lines of the introduction of autoregressive moving average (ARMA) and generalized autoregressive and conditional heteroscedasticity approaches (GARCH) to economic data (see Bollerslev (1986) and Engle et al. (2008)), we discuss the ability of ML and DL to capture autocorrelation structure and heteroscedasticity of financial returns and also address the accuracy of ML when the number of historic data points is limited. In doing so, we shed light on the underlying mechanism of ML and DL, which is highly

This is an open access article under the terms of the [Creative Commons Attribution](https://creativecommons.org/licenses/by/4.0/) License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

© 2025 The Author(s). *Journal of Forecasting* published by John Wiley & Sons Ltd.

relevant for financial risk measurement and management. Hence, we provide an innovative in-depth assessment of ML and DL beyond the widely accepted ML metrics such as in-sample accuracy and out-of-sample performance. We assess both simulated ARMA–GARCH processes and empirical data and analyze, based on residuals, if the performance of neural networks and deep neural networks is driven by their ability to adequately capture linear time series properties or by hidden information which are not revealed by the first two moments of a time series.

Our study adds to Gu et al. (2020). The authors provide an in-depth assessment of neural networks to economic cross-sectional data and find that ML describes a promising alternative for adequate return prediction. We also assess daily financial data, however, in contrast to Gu et al. (2020); our focus lies on univariate time series assessment. That is, the explanatory variables in our study are given by past observations with limited horizon. Furthermore, we address the current studies dealing with LSTM-based forecasting techniques as summarized in Makridakis et al. (2018). We add to this string of literature by explicitly discussing econometric ARMA processes via ML. This enables us to study the financial properties which are captured via ML, based on the statistical properties of the residuals. In doing so, we provide a thorough benchmark study for daily return forecasting.

The contribution of our paper is threefold. We provide an in-depth assessment of economic time series analysis of daily financial return series via ML and present a pioneering benchmark study for economic hyperparameter tuning. Second, we evaluate competing ML approaches by their ability to capture economic stylized facts, such as autoregression and heteroscedasticity. Third, we identify the optimal configuration of ML technique, which outperforms traditional ARMA approaches. Based on simulated and empirical data sets, we find that plain vanilla recurrent neural networks (RNNs) describe an adequate alternative to econometric ARMA approaches. The remainder of this paper is structured as follows. Section 2 provides the relevant literature on time series analysis and ML, Section 3 gives an overview on the methodology and Section 4 presents the results of the simulation study. The investigated data and the empirical assessment are in Section 5, and Section 6 concludes the study.

## 2 | Literature

As described in Engle et al. (2008), univariate financial time series (i.e. periodically returns) exhibit asset specific properties, such as heteroscedasticity and autocorrelations. Due to its translation into expected returns and financial risk, it is the first and second moments of the return distribution that describes the balance point of econometric modeling. Moreover, these moments build the connection between econometric time series analysis and financial applications such as Value-at-Risk forecasts or portfolio allocations (see Jorion (2007) and Verbeek (2017)). In this vein, ARMA approaches as introduced in Bollerslev (1986) and GARCH as introduced in Engle et al. (2008) present the widely accepted benchmarks in

recent literature (see Berger and Gencay (2018), Halbleib and Pohlmeier (2012)).

In contrast to econometric modeling, ML approaches are less restrictive regarding the assumptions on the input variables and are also able to handle larger data sets. Triggered by Cochrane (2011), recent focus lies on the application of sophisticated deep neural networks to economic data. However, due to complexity, ML and DL are characterized as black-box approaches, and the superiority of these approaches in comparison with existing statistical approaches is often assessed via the ability to provide an adequate fit to the underlying target variable, measured in terms of the precision of in-sample and out-of-sample accuracy (see Avramov et al. (2022) and Gu et al. (2020)). With focus on financial time series data, it is the limited amount of historic data that describes the main challenge for the application of ML to financial time series. Up to now, mixed results are reported for individual financial time series. It is widely accepted that RNNs describe the adequate network structure to handle timely dependent data (Makridakis et al. (2018)). Giles et al. (2001) apply neural networks to forecast five different daily foreign exchange rates (German DM, JPY, Swiss Franc, GBP and CAD denominated against USD) from 1973 to 1987 and explicitly focus on the aspect of nonstationarity, overfitting, and time persistent mean. The authors demonstrate that RNN describes an adequate choice for financial time series. Krauss et al. (2017) investigate the potential of arbitrage via ML. The authors assess daily data of all stocks which were listed in the S&P 500 from 1992 to 2015 via ML, that is, random forest, gradient boosted trees, deep neural networks, and ensembles of these methods. As a result, the authors empirically demonstrate that an ensemble approach provides superior performance in terms of out-of-sample returns. However, the authors do not apply memory-based ML approaches.

Bhandari et al. (2022) address this gap in the literature and assess daily S&P 500 market prices and demonstrate that single layer LSTM frameworks perform better than multilayer LSTM models with two or three layers. This finding is also confirmed in Yadav et al. (2020) for the Indian stock market data. Furthermore, as noted in Fjellström (2022), there are only a few studies which address a comparison between both ML and econometric methods. Vortelinos (2017) assesses NN in comparison to heterogeneous autoregressive models and principal component analysis. Mixed evidence is reported; however, plain-vanilla neural networks are outperformed by RNNs and LSTM networks. Siami-Namini et al. (2018) compare the forecasting performance of LSTM networks against ARMA approach for monthly returns of Asian and American indices (Nikkei 225, Hang Seng Index, Nasdaq Composite, Dow Jones Industrial Index, and S&P 500 Commodity Price Index). The authors find that LSTM approach provides a reduction of in-sample RMSE up to 80%. Interestingly, the data sets are quite small (due to monthly data) and comprise between 258 and 1189 observations; however, the authors train only one epoch. Based on the forecasting competition initiated by Makridakis and Hibon (2000), Makridakis et al. (2020) provide a thorough summary on recent forecasting challenges and point out that there exists no approach which categorically leads to superior forecasting accuracy.

### 3 | Methodology

In this section, we introduce the notation used throughout the paper; we define the desirable properties of the training and test data, specify the econometric benchmark model and RNNs, and present our performance evaluation settings.

#### 3.1 | Generating Training and Test Data

Based on daily market prices, we calculate daily financial return series. Let the market price of an asset at time  $t$  be defined as  $P_t$  with  $t = 1, \dots, T$ , then the daily return is calculated as follows:

$$r_t = \frac{P_t}{P_{t-1}} - 1 \quad (1)$$

In order to assess both in-sample and out-of-sample precision of the competing approaches, we follow Krauss et al. (2017) and define nonoverlapping training and test periods. The training period comprises daily returns of 1.024 consecutive days (approximately 4 years) and is split into two data sets: the first 768 days describe the training data, and the last 256 days are the evaluation data set. The training period is the underlying data to estimate the parameters of econometric benchmark and to train the RNNs. The test data period comprises the following 256 consecutive days and allows us to test the models with data, which was not part of the estimation or training process. In doing so, the training period serves as the basis for the in-sample assessment, and the test period describes the returns of the out-of-sample assessment.

#### 3.2 | Econometric Benchmark

As discussed in Engle et al. (2008), econometric time series analysis via ARMA approach takes into account economic properties of daily financial return series, namely, autocorrelation and heteroscedasticity of financial returns. The ARMA approach of order  $(p, q)$  for daily returns is defined as follows:

$$r_t = \phi_0 + \sum_{i=1}^p \phi_i r_{t-i} + \sum_{j=1}^q \theta_j \epsilon_{t-j} + \epsilon_t \quad (2)$$

Here,  $\phi_0$  is a constant, and if the process is weakly stationary,  $\epsilon_t$  is a white noise process with mean 0 and  $\sigma_t^2$ .  $p$  and  $q$  describe the lags of the past returns  $r_{t-i}$  and  $\epsilon_{t-j}$ , respectively. Due to the MA term, ARMA models must be estimated using *maximum likelihood estimation*. Typically, as described in Engle et al. (2008), economic time series are well described by ARMA(1,1) approaches. In our study, we examine competing ARMA approaches for each time series and assess the optimal combination of  $p$  and  $q$  with up to five lags.

Furthermore, in our simulation setup, we simulate realistic returns that are also characterized by time-varying volatility clustering via the GARCH (1,1) approach as defined by Bollerslev (1986). The model is given by the following:

$$\sigma_t^2 = \Omega + \alpha r_{t-1}^2 + \beta \sigma_{t-1}^2; \quad (3)$$

here,  $\Omega$  is a constant,  $\alpha$  accounts for the news impact, and  $\beta$  describes the persistence of the conditional volatility process, with  $\alpha + \beta \leq 1$ .

#### 3.3 | RNNs

For a detailed in-depth description of the applied ML approaches, we refer to Goodfellow et al. (2017) and Russell et al. (2010). However, as described in LeCun et al. (2015) and Krauss et al. (2017), RNNs describe the adequate ML approach for time series. In contrast to econometric approaches, input data for ML need to be standardized. We draw on Krauss et al. (2017) and normalize the daily returns before they get passed to ML exercise. The input for network training is then defined as follows:

$$y_t = \frac{r_{t,train} - \mu_{train}}{\sigma_{train}} \quad (4)$$

Here,  $r_{t,train}$  describe the returns of the training period at time  $t$ ,  $\mu_{train}$  is the mean return of the training period, and  $\sigma_{train}$  is the respective standard deviation. In order to predict future returns of a time series, ML models take transformed input and target returns as training data in order to identify functional relationships and to predict  $\hat{y}_t$ . Furthermore, in order to compare the predicted output, predicted returns are then constructed via

$$\hat{r}_t = \hat{y}_t \cdot \sigma_{train} + \mu_{train} \quad (5)$$

It is necessary to take the mean and standard deviation from the training set to ensure that there is no look-ahead bias. Similar to econometric time series approach, past returns describe the only input for model training, and along the lines of the ARMA approach, we determine past returns as the input variables, that is, the feature vector and the  $k$ -step ahead return as the target variable. Then, the applied ML approaches can be boiled down to the following equation:

$$\hat{y}_{t+k} = f_k(y_{t-w}, \dots, y_{t-1}, y_t) \quad (6)$$

with  $\hat{y}_{t+k}$  as the forecast of the target variable for time  $t+k$  at time  $t$  and  $k$  is the forecasting horizon.  $y_{t-w}, \dots, y_t$  are the past observations of the target variable, observed from time  $t-w$  until  $t$  and  $w$  are the window sizes. The target variables are normalized daily returns, and the input variables are lagged observations of the target variables. In our study, we focus on one-day ahead forecasts ( $k=1$ ), and  $f_k$  is the trained ML function, and we study RNNs, long short-term memory (LSTM), and gated recurrent units (GRUs). Then, in combination with Equation (4),  $\hat{r}$  can be constructed.

##### 3.3.1 | RNN

RNNs are a specific type of artificial neural networks and capture lagged correlation structure between the input and target variables. The network preserves relevant sequential information in an inner (hidden) state, which persists subsequent time steps. During the training of the network, the RNN uses the input variables at time  $t$  and the preserved past information from previous step  $t-1$  as the input for target variable at step

$t$ . Then, based on a training data set which consists of historical data, RNN recurrently learns the structural relationship between input and target variables by modeling the transitions from inner state at  $t - 1$  to  $t$ . Hence, in contradiction to simple neural networks, RNNs share the same set of weight parameters during the training process.

Specifically, as described in Gu et al. (2020), the resulting RNN comprises three parameter matrices:  $W_x$ ,  $W_y$ , and  $W_s$  and two bias vectors  $b_s$  and  $b_y$ . The output at time  $t$  ( $y_t$ ) is then determined by the inner (hidden) state at time  $t$  ( $S_t$ ) which is determined by input at time  $t$  ( $x_t$ ) and the previous hidden state ( $S_{t-1}$ ). In the remainder,  $x_t$  exclusively comprises lagged versions of the target variable, and the setup of the RNN can be summarized as follows:

$$S_t = \tanh(W_{xs} \cdot (x_t \oplus S_{t-1}) + b_s) \quad (7)$$

and

$$y_t = \sigma(W_y \cdot S_t + b_y) \quad (8)$$

Here,  $\sigma$  is the sigmoid activation function,  $\tanh$  is the nonlinear  $\tanh$ -function, and  $x_t \oplus S_{t-1}$  is the concatenation of the two vectors  $x_t$  and  $S_{t-1}$ . Due to the repeated multiplication of recurrent parameter matrices, RNNs often have difficulties in learning long-term trends from past information and often capture sequential information with relatively short memories. As described in Bengio et al. (1994), this finding is caused by constraints in gradient based learning algorithms. As a result, RNNs might fail in capturing persistent long-run dynamics of financial time series.

### 3.3.2 | LSTM

To overcome the issue of vanishing gradients, Hofreiter and Schmidhuber (1997) introduced the LSTM architecture. In contradiction to plain vanilla RNN, the hidden layer neurons get replaced by a memory cell which is designed to capture dependence between long-term dependence between lagged periods. The LSTM memory cell includes an input gate  $i$ , a forget gate  $f$ , and an output gate  $O$ .

The input for the forget gate  $f$  at time  $t$  is the output from the previous time step ( $S_{t-1}$ ) and the input variables at time  $t$ . Then, the forget gate controls for the information of the cell state which will be passed from the previous period to the next period. The values  $f$  range between 0 and 1, whereas the limits indicate that the complete information will be passed or discarded, respectively.  $f$  is then given as follows:

$$f_t = \sigma(W_f \cdot (x_t \oplus S_{t-1} + b_f)) \quad (9)$$

The input gate controls for the information from input  $x_t$  that will be passed to the current memory cell  $C_t$ , and the rest will be erased for the next period. Updating the new cell state  $C_t$  can be described as follows:

$$i_t = \sigma(W_i \cdot (x_t \oplus S_{t-1} + b_i)) \quad (10)$$

$$\tilde{C}_t = \tanh(W_C \cdot (x_t \oplus S_{t-1}) + b_C) \quad (11)$$

and

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \quad (12)$$

First, the relevant information  $i_t$  is generated, and a newly cell state  $\tilde{C}_t$  is generated based on previous cell state. Then, both components are merged, as described in Equation (12), and built the updated current cell state.

The third gate, the output gate, determines how much of the newly generated cell state will be passed to the next period:

$$O_t = \sigma(W_O \cdot (x_t \oplus S_{t-1} + b_O)) \quad (13)$$

$$S_t = \tanh(C_t) \cdot O_t \quad (14)$$

Then, the processed cell state and the sigmoid output build  $S_t$  which is then processed to the final output via

$$y_t = \sigma(W_y \cdot S_t + b_y) \quad (15)$$

As a result, the LSTM architecture allows for a more sophisticated memory structure which explicitly allows for learning longer-term dependencies between lagged periods. Långkvist et al. (2014) demonstrate that LSTM networks process the ability to remember 100 steps of a sequence.

### 3.3.3 | GRUs

Another alternative variant to simple RNN is the GRU, which can also be described of as a variant of LSTM. GRU networks process information through an update gate ( $z$ ) and a reset gate ( $re$ ) plus an additional gate  $\tilde{S}_t$ . The functions  $re_t$  and  $z_t$  are given as follows:

$$re_t = \sigma(W_r \cdot (x_t \oplus S_{t-1}) + b_r) \quad (16)$$

and

$$z_t = \sigma(W_z \cdot (x_t \oplus S_{t-1}) + b_z) \quad (17)$$

and the third gate generates a new cell state  $\tilde{S}_t$  by combining the new input at time  $t$  with previous cell state  $S_{t-1}$  in combination with the reset gate:

$$\tilde{S}_t = \tanh(W_s \cdot (x_t \oplus S_{t-1} \cdot r_t) + b_s) \quad (18)$$

Then, the update gate controls for the information which is retained from previous state ( $S_{t-1}$ ) as well as from the generated state  $\tilde{S}_t$ , which is then stored in the new cell state ( $S_t$ ):

$$S_t = (1 - z_t) \cdot S_{t-1} + z_t \tilde{S}_t \quad (19)$$

Analogous to RNN and LSTM, the information is then processed to the output  $y_t$

$$y_t = \sigma(W_y \cdot S_t + b_y) \quad (20)$$

In comparison to LSTM, GRU is described by less complexity and is therefore less time-consuming when it comes to model

training. As discussed in Shi (2022), both LSTM and GRU do co-exist.

### 3.3.4 | Model Training

The presented networks are trained, based on the training period which consists of consecutive training and validation subsample. We follow Fischer and Krauss (2018) and Granger (1993) and use 75% as training subsample (approximately three years of daily data). The trained model weights are then assessed on the validation subsample (approximately 1 year of daily data), and the validation loss is computed after each epoch. The training and validation loss are then the indicator for the next training epoch.

Table 1 summarizes the different scenarios that will be assessed in our study. In order to include the presented configurations of the quoted literature, we assess different architectures in order to study the impact of complexity on the forecasting precision.<sup>1</sup> We study different numbers of hidden neurons. Specifically, in order to compare the amount of parameters to the ARMA approach, we start with five neurons for each network and increase the number up to 100 neurons. Also, we assess the impact of network depth and study the impact of additional layers on the forecasting precision. Starting from one layer, we increase the size up to four layers.

Furthermore, for the training of the networks, we follow Gal and Ghahramani (2016) and Fischer and Krauss (2017) and implement dropout regularization within the recurrent layer. That is, at each update, a percentage of input units is randomly dropped out at each update of the training process at both input gate and recurrent connections. We assess different drop out rates, ranging from 0% to 0.2%; in doing so, we avoid overfitting. Furthermore, we study the impact of epochs on the model accuracy; we assess different scenarios of model architectures as well as training processes. For each network architecture, we assess 19 different scenarios of training epochs with a maximum duration of 1000 epochs. Consequently, we discuss 9120 competing network configurations.

TABLE 1 | Topology of networks.

Training scenarios	
Architecture	Simple RNN, LSTM, GRU
Neurons	5, 25, 50, 75, 100
Layer	1, 2, 3, 4
Dropout rate	0.0, 0.1, 0.2
Batch size	2, 4, 8, 16, 32, 64, 128, 256
Epochs	10, 20, 30, ..., 100, 200, 300, ..., 1000
Loss function	MSE
Optimizer	adam, rmsprop (Keras)

Note: The table presents the configuration parameters of the recurrent neural networks that are assessed in our study. RNN is the simple recurrent neural network, LSTM is long short-term memory, and GRU is gated recurrent units.

## 3.4 | Performance Evaluation

In addition to typical assessment of neural nets, we focus on the generated return series of both approaches and discuss the forecasting accuracy via root mean squared error (RMSE) and assess the autocorrelation of the residuals via the Ljung–Box (LB) test. Also, as we assess various ML architectures, we identify statistical differences between competing forecasts via the Diebold–Mariano (DM) test statistic.

### 3.4.1 | RMSE

In order to compare both econometric and ML approaches, we assess the RMSE of the predicted returns against the observed returns:

$$RMSE = \sqrt{\frac{1}{n} \sum_{t=1}^T (r_t - \hat{r}_t)^2} \quad (21)$$

Instead of looking at the difference between the output  $\hat{y}$  of each ML approach, we assess the artificial returns and which enables us to compare ML performance against the generated returns from econometric benchmark.

### 3.4.2 | LB

In addition to the assessment of forecasting precision, we apply the LB tests, to study the residuals of each model. Let the residual of model  $a$  at time  $t$  be defined via

$$u_t^{(a)} = r_t - \hat{r}_t^{(a)} \quad (22)$$

Then, the diagnostics on the residuals can be investigated via LB test. The null hypothesis is that residuals are independently distributed over time, and the test statistic is given as follows:

$$LB(m) = T(T+2) \sum_{t=1}^m \frac{\hat{\rho}_t^2}{T-t} \sim \chi_m^2 \quad (23)$$

with  $\chi_m^2$  as the chi-squared distribution with  $m$  degrees of freedom.

### 3.4.3 | The DM Test

Due to the fact that we assess various combinations of hyper-parameters, we assess the pairwise difference between different forecasts via the DM test as described in Diebold and Mariano (1995). The difference between two models, lets say model  $a$  and model  $b$ , at time  $t$  can then be assessed via respective forecasting errors:

$$d_{ab,t} = g(u_t^{(a)}) - g(u_t^{(b)}) \quad (24)$$

with  $g(\cdot)$  as the squared-error loss function and  $u_t^{(a)}$  and  $u_t^{(b)}$  are calculated as presented in Equation (23). Then, based on

the loss-differential series, we test the null hypothesis that two forecasts are characterized by same accuracy, that is,  $E(d_t) = 0$  for  $t = 1, \dots, T$ , and the test statistic between approach  $a$  and approach  $b$  is given as follows:

$$DM_{a,b} = \frac{\bar{d}_{ab}}{\sqrt{\frac{2\pi f_d(0)}{T}}} \quad (25)$$

with  $\bar{d}_{ab} = \frac{1}{n} \sum_{t=1}^T (d_{ab,t})$  is the mean of the loss differential and  $f_d(0)$  is the density of the loss differential at frequency 0. Under the assumption that the loss differential series is covariance stationary, the test statistic follows a standard normal distribution. For an in-depth introduction, we refer to Diebold and Mariano (1995) and for a critical discussion on the validity of the underlying assumptions, especially on covariance-stationary errors we refer to Diebold (2015).

## 4 | Simulation Study

In this section, we introduce the concept of simulated return processes and provide an in-depth assessment of the optimal choice of neural net for time series, which are characterized by financial stylized facts, namely, AR, MA, and GARCH. The starting point of our simulation assessment builds a simple neural net with the same amount of parameters (neurons) like typical econometric benchmarks. We show that additional complexity does not necessarily result in more precise time series modeling. In addition to modeling accuracy, we also study the residuals in order to gain further insights into the autocorrelation structure that is captured by competing ML approaches.

### 4.1 | Return Process

Generally, financial return processes are adequately characterized via the ARMA-GARCH processes. In order to simulate realistic return series, we fit the ARMA-GARCH models to daily return series and take the estimated parameters as the input parameters for our simulation assessment. The stationary process, which is then discussed in the remainder, is described as follows:

$$\hat{r}_{t|sim} = \phi_0 + \phi_1 r_{t-1|sim} + \theta_1 \epsilon_{t-1|sim} + \epsilon_t \quad (26)$$

$$\hat{\sigma}_{t|sim}^2 = \Omega + \alpha r_{t-1|sim}^2 + \beta \sigma_{t-1|sim}^2 \quad (27)$$

Here,  $\phi_1$  equals 0.8 and describes the autocorrelation structure of the conditional mean and the  $\theta_1$  is 0.1 and describes the moving average effect, namely, the impact of the previous residual on the current simulated returns. Additionally, we model conditional variance via  $\beta$  equal to 0.94 and  $\alpha$  equal to 0.04. Hence, the modeled process is similar to volatility processes as described by *RiskMetrics<sup>TM</sup>* and results in a volatility process which is characterized by volatility clustering. For our simulation study, we simulate return series with 1280 observations which is similar to 5 years of daily return data.<sup>2</sup>

Figure 1 illustrates the simulated return series as well as the partial autocorrelation function of the returns and the squared returns respectively. Obviously, autocorrelation is present in both conditional mean and conditional volatility process, and we will discuss the ability of the competing ML approaches based on both accuracy and residuals. Also, in order to discuss both in-sample and out-of-sample performance, similar to Krauss et al. (2017), we split the data into a training and test data set. The training data set covers 4 years (1024 days) and builds the basis for model training and validation. Therefore, 0.75% of the training data deal as the testing and 0.25% of the training data are determined for model validation. The test data set comprises the fifth year (256 days) and allows us to evaluate a strict out-of-sample performance of the applied models.

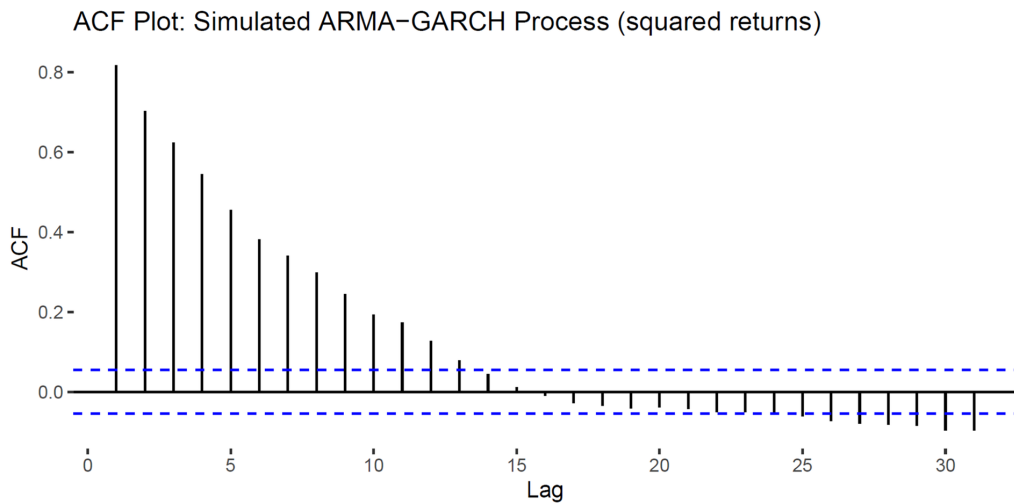
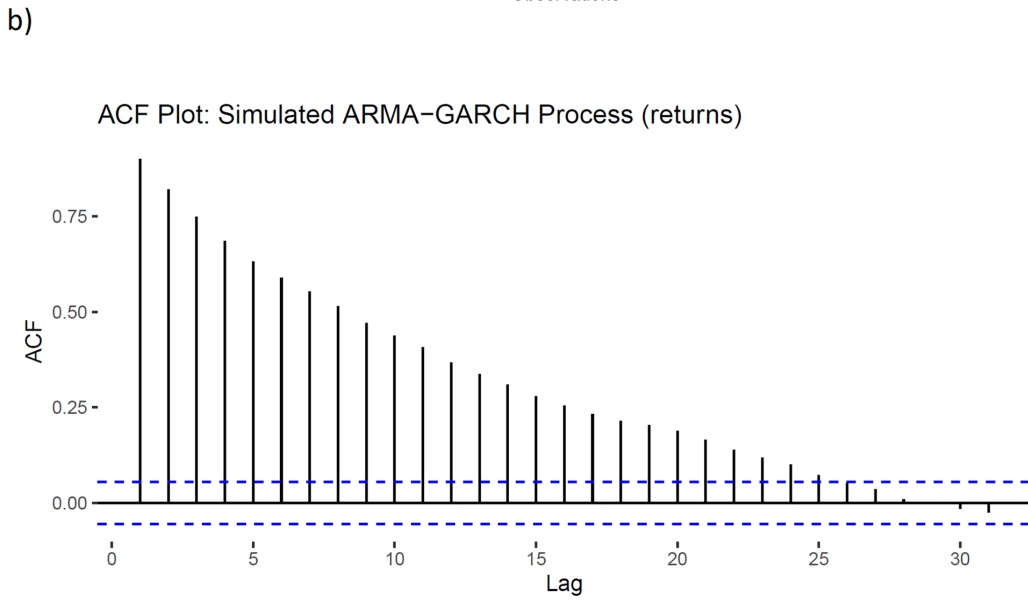
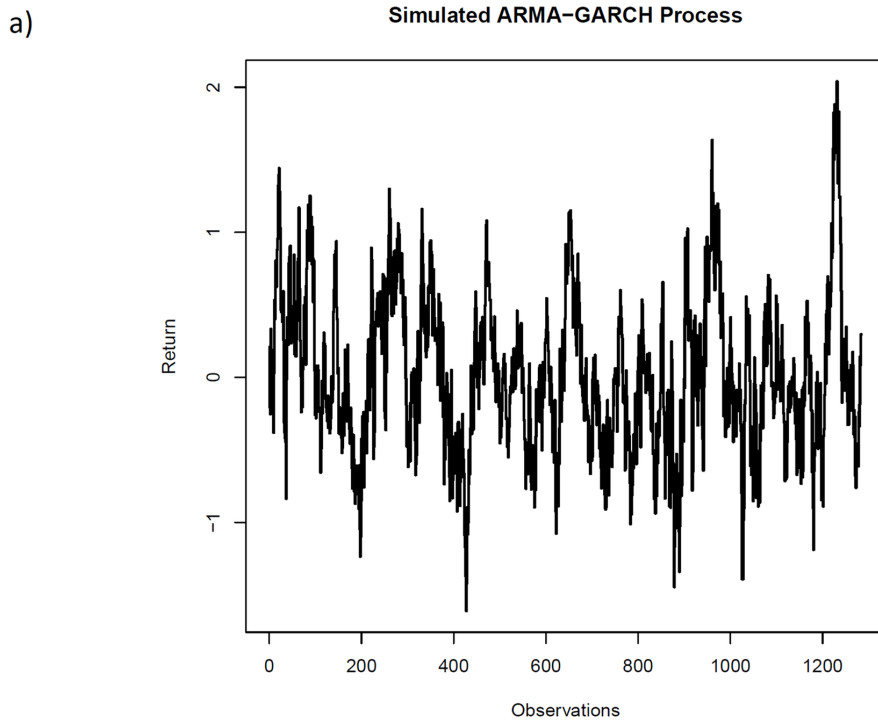
### 4.2 | ML Assessment

In order to study the ability of ML techniques to handle economic time series, we begin the analysis by studying a plain vanilla RNN with similar amount of parameters like the ARMA-GARCH approaches. Hence, a RNN with one layer and five neurons describes the starting point of our assessment. In contradiction to econometric approaches, parameters are not estimated via maximum likelihood approach, and therefore, we begin with hyperparameter tuning and focus on the identification of the optimal amount of training epochs (i.e., one epoch describes the training cycle until all data is processed once) and the optimal batch size which is the amount of observations which are presented to the model at the same time. Hence, the individual steps within one epoch vary by the dimension of batch size. To avoid in-sample overfitting, we explore different dropout rates.

Thus, we begin our assessment and study in-sample and out-of-sample performance of the RNN with one layer and five neurons for different batch sizes and epochs. The batch size varies from 2 to 256 observations, which describes the maximum in our study and which is similar to one year of data, which is processed at one step. The amount of assessed epochs begins with 10 and ends with 1000. For each combination of epochs and batch size, the in-sample and out-of-sample RMSE is scrutinized. The LB test statistic of the residuals is also assessed to study the autocorrelation function. As a result, we find that the in-sample RMSE increases with increasing batch size; that is, the more data are processed at once, the less precise the in-sample prediction gets. Furthermore, we find that an increasing amount of epochs leads to a higher precision for both in-sample and out-of-sample RMSE. The optimal combination in this exercise is achieved with 100 epochs and a batch size of 32 observations. There is no other combination which leads to lower RMSE for both subsamples. Furthermore, as indicated by the LB test statistic, simple RNN is able to capture the generated autocorrelation structure; for nearly all combinations, residuals are not characterized by autocorrelation up to lag 20.

#### 4.2.1 | Hyperparameter Tuning

Taking the simple RNN with five neurons as a starting point, we also discuss expansions of RNN and compare the results



**FIGURE 1** | Legend on next page.

**FIGURE 1** | Simulated returns over time. (a) A time series with 1280 observations is generated via the ARMA–GARCH approach with  $r_{t|sim} = 0.8 \cdot r_{t-1|sim} + 0.1 \cdot \epsilon_{t-1|sim}$  and  $\sigma_{t|sim}^2 = 0.04 \cdot r_{t-1|sim}^2 + 0.94 \cdot \sigma_{t-1|sim}^2$ . (b) The autocorrelation of the simulated returns as well as the squared simulated returns.

against the identified optimal combination. Specifically, we add complexity to the layers by adding more neurons, and we add complexity to the network architecture by adding more layers. As well, we discuss more sophisticated layer structures, namely, GRU and LSTM layers. All results will additionally be assessed via DM Test statistic.<sup>3</sup>

**Impact of additional nodes:** We find that with increasing complexity, the amount in-sample and out-of-sample precision increases. For RNNs with 50, 75, and 100 nodes, the optimal combination is given by large batch size (256) and 1000 epochs, whereas RNN with 50 nodes describes the optimal choice for the underlying data. Specifically, there is no amount of nodes and no combination of batch size and epochs that result in lower RMSE for both in-sample and out-of-sample jointly. As indicated by LB test statistic, residuals do not exhibit autocorrelation and DM test statistic indicates that predictions based on RNN with 50 neurons statistically differ from RNNs with five neurons. Hence, we find that 50 neurons describe the optimal choice for an RNN with one layer and we carry on, by assessing the impact of added layers, namely, the impact of DL on the forecasting precision.

**Impact of additional layer:** In comparison to the performance of a simple RNN with one layer and 50 neurons, we find that adding additional layers to the RNN does not improve the forecasting performance. Specifically, there is no model which leads to lower in-sample and out-of-sample performance. With focus on PACF, we find that the  $p$ -values of LB test statistic decrease with each layer, and we find that the residuals of a RNN with three and four layers are characterized by autocorrelation for batch size 2. The results indicate that adding additional layers to a RNN does not result in adequate economic time series forecasts. Also, the fact that residuals exhibit autocorrelation indicates that time series specific properties are not adequately captured with additional layers.

#### 4.2.2 | GRUs and LSTM

We also assess the performance of a more complex network architecture and analyze the time series forecasting based on GRU and LSTM layers. That is, we apply plain vanilla networks with both one GRU or LSTM layer and assess the impact of different nodes (10, 50, and 100). Analogous to the assessment of simple RNN, we also assess the impact of additional layers on the forecasting performance. The results are compared to RNN with layer and 50 neurons.

With focus on architectures with one layer, we find that the application of GRU networks marginally outperforms LSTM networks with same amount of neurons in terms of in-sample RMSE, whereas LSTM provides a marginally lower out-of-sample RMSE. For both approaches, additional neurons increase both in-sample and out-of-sample precision.

The minimum in-sample RMSE decreases from 0.3006 (with 25 neurons) down to 0.2887 (with 100 neurons) for LSTM approach and from 0.2989 (with 25 neurons) to 0.2888 (with 100 neurons). The LSTM out-of-sample RMSE decreases from 0.3094 (with 25 neurons) to 0.3053 (with 100 neurons) and the GRU RMSE from 0.3118 to 0.3068, respectively. For both approaches, it is the batch size which impacts the balance between in-sample and out-of-sample RMSE. That is, an increasing batch size increases the in-sample RMSE and decreases the out-of-sample RMSE.

We assess in-sample and out-of-sample RMSE for different combinations of layers. We find that the impact of additional layers is also similar for both LSTM and GRU approach, adding a layer to the model architecture results in lower in-sample RMSE but does not impact out-of-sample RMSE. Specifically, for LSTM networks, the minimum of in-sample RMSEs decreases from 0.2887 (one layer with 100 nodes) to 0.2066 (four layers with 50 nodes each), whereas out-of-sample RMSE increases from 0.3053 to 0.3171, respectively. Analogously, in-sample RMSE of GRU networks increases from 0.2887 to 0.2064, and out-of-sample RMSE increases from 0.3086 to 0.3127. Hence, we find that additional complexity leads to improved in-sample fit; however, one layer with 50 to 75 neurons and a batch size equal to 256 describes an adequate choice for the simulated returns. At this point, there is no combination which results in both lower in-sample and out-of-sample RMSE. However, both models get outperformed by plain vanilla RNN with 50 neurons. Both LSTM and GRU networks result in higher in-sample and out-of-sample RMSE per se. Furthermore, as indicated by LB test, the resulting residuals exhibit autocorrelation, and therefore, both approaches are not able to adequately capture the linear autocorrelation structure of the simulated ARMA–GARCH time series.

Table 2 provides a summary of the conducted simulation exercise. For each competing ML approach, the model network architecture which leads to the lowest in-sample RMSE, the lowest out-of-sample RMSE, and a balanced performance is presented. The balanced performance describes the most simple architecture which leads to a combination of lowest in-sample and out-of-sample RMSEs jointly, which are not outperformed by another more complex configuration of the same network approach. That is, a RNN with one layer and 50 neurons provides a combination of in-sample and out-of-sample RMSE (0.2296 and 0.2534), and there is no RNN approach that results in a higher accuracy for both in-sample and out-of-sample jointly. In comparison to LSTM and GRU, RNN describes the higher accuracy for both training and test data. Also, as indicated by LB test statistic, RNN is the only approach which results in residuals that are not characterized by autocorrelation structure. That is, RNN is the only approach that captures the generated, asset specific, linear ARMA–GARCH autocorrelation structure. Also, the lowest in-sample and out-of-sample RMSE which can be achieved by each network

TABLE 2 | Simulation study.

Scenario	Layer	Units	Batchsize	Epochs	In-sample			Out-of-sample
					RMSE	LB	DM	RMSE
Balanced								
RNN	1	50	256	1000	0.2296	0.18	X	0.2534
LSTM	1	75	256	200	0.3081	0.00	0.00	0.3068
GRU	1	50	256	200	0.3089	0.00	0.00	0.3097
Lowest in-sample								
RNN	4	50	16	1000	0.2196	0.29	0.18	0.2556
LSTM	4	50	8	1000	0.2066	0.00	0.00	0.3933
GRU	4	50	8	800	0.2063	0.00	0.00	0.3932
Lowest out-of-sample								
RNN	1	75	256	500	0.2299	0.13	0.06	0.2526
LSTM	1	100	256	200	0.3053	0.00	0.00	0.3087
GRU	1	100	256	100	0.3068	0.00	0.00	0.3120

Note: The table presents the root mean squared error (RMSE) of the in-sample and out-of-sample prediction accuracy of the best performing simple recurrent neural nets (RNN), long short-term memory (LSTM), and gated recurrent unit (GRU) networks with 5, 25, 50, 75, or 100 neurons and one, two, three, or four layers. For each simulation scenario, the optimal combination of neurons, batch size, and number of epochs is presented. The  $p$ -values of the Ljung-Box (LB) test statistic indicate the existence of autocorrelation of the residuals with  $H_0$ . Residuals do not exhibit autocorrelation. The  $p$ -values of the Diebold-Mariano (DM) test statistic indicate the statistical difference between the predicted values of network and a plain vanilla RNN with 50 neurons with  $H_0$ . The generated time series are statistically similar.

approach indicates that RNN provides a stable performance among all tested architectures.

### 4.3 | Robustness

A note on robustness: Similar to Krauss et al. (2017), we have repeated the model training three times. As well, we have assessed different simulated ARMA and GARCH processes to study scenarios that are described by stronger and weaker degrees of autocorrelations, and we have varied the dropout rate to assess the impact of overfitting on our results.

Table 3 gives the assessed scenarios. Scenarios 1–5 describe scenarios with different autocorrelation structure in the generated conditional mean, scenarios 6–9 are scenarios with different autocorrelation structure in the time varying conditional variance process, and scenarios 10–14 describe scenarios with different dropout rates during model training. The finding that plain vanilla RNNs outperform sophisticated LSTM and GRU approaches is robust; for all scenarios, we find that LSTM and GRU networks get outperformed by both, in-sample and out-of-sample RMSE of the predicted returns. Moreover, the DM test statistic indicates that the in-sample fit via RNN is statistically different to GRU and LSTM. Although the simulated processes are described by linear autocorrelation, we find that both GRU and LSTM are not capable of producing uncorrelated residuals. The dropout rate does not impact the superior performance of RNN; however, if we train the model without a dropout factor (dropout=0), RNN do also provide in-sample residuals which exhibit autocorrelation. That indicates the potential impact of overfitting, when no dropout is applied via the training process.

## 5 | Empirical Assessment

The data comprise daily return series of the S&P 500 for the period ranging from January 1, 2012 to December 31, 2022. We split the full sample into five subsamples to investigate the out-of-sample performance for 2018, 2019, 2020, 2021, and 2022. Each subsample is then divided into a training and validation period; that is, the basis for the in-sample assessment and the test data set describes the underlying information for the out-of-sample period.<sup>4</sup>

Table 4 provides descriptive statistics of the five investigated subsamples of the daily DJI return series as well as the fitted ARMA parameters. We can see that the periods covering the COVID-crisis in 2020, 2021, and 2022 are characterized by higher standard deviations than in 2018 and 2019. Specifically, in 2022, the negative average return is four times higher than in 2018 and the standard deviation increased by 56% in comparison to 2019. The division of the subsamples enables us to analyze the ability of competing approaches to anticipate market regimes that are different from the training data, and we study scenarios, in which market reactions on COVID become part of the training data. We study the past 11 years of daily returns from of S&P 500. We split the data into a training and validation set comprising 1024 days and a test data set comprising 256 days. In doing so, we train the competing ML approaches based on 4 years of daily returns and study both the in-sample forecasts and the out-of-sample forecasts based on the test data set. We retrain each model after 1 year, which allows us to study 5 years of out-of-sample forecasts, namely, January till December 2018, 2019, 2020, 2021, and 2022. Furthermore, we assess the same network architectures as presented in Table 1 for each individual model specification; 80 different combinations of batch size

TABLE 3 | Simulation study: Overview on robustness assessment.

Robustness	ARMA			GARCH				RNN			LSTM			GRU									
	$\phi_0$	$\phi_1$	$\theta$	$\Omega$	$\alpha$	$\beta$	Dropout	RMSE	IS	LB	RMSE	OOS	IS	LB	DM	RMSE	IS	LB	DM	RMSE	OOS		
Scenario 1	0.005	0.2	0.7	0.001	0.04	0.94	0.2	0.2370	0.00	0.00	0.2370	0.3196	0.00	0.00	0.00	0.3221	0.3165	0.00	0.00	0.3217	0.3165	0.00	0.3217
Scenario 2	0.005	0.4	0.5	0.001	0.04	0.94	0.2	0.2313	0.00	0.00	0.2617	0.3175	0.00	0.00	0.00	0.3158	0.3157	0.00	0.00	0.3134	0.3157	0.00	0.3134
Scenario 3	0.005	0.6	0.3	0.001	0.04	0.94	0.2	0.2293	0.26	0.2567	0.2567	0.3141	0.00	0.00	0.00	0.3089	0.3128	0.00	0.00	0.3089	0.3128	0.00	0.3089
Scenario 4	0.005	0.8	0.1	0.001	0.04	0.94	0.2	0.2292	0.25	0.2527	0.2527	0.3089	0.00	0.00	0.00	0.3054	0.3080	0.00	0.00	0.3069	0.3080	0.00	0.3069
Scenario 5	0.005	0.9	0.05	0.001	0.04	0.94	0.2	0.2298	0.25	0.2530	0.2530	0.3160	0.00	0.00	0.00	0.3174	0.3165	0.00	0.00	0.3226	0.3165	0.00	0.3226
Scenario 6	0.005	0.8	0.1	0.001	0.24	0.74	0.2	0.2013	0.03	0.2563	0.2563	0.2662	0.00	0.00	0.00	0.2911	0.2650	0.00	0.00	0.2908	0.2650	0.00	0.2908
Scenario 7	0.005	0.8	0.1	0.001	0.44	0.54	0.2	0.1265	0.09	0.1839	0.1839	0.1674	0.00	0.00	0.00	0.1950	0.1668	0.00	0.00	0.1945	0.1668	0.00	0.1945
Scenario 8	0.005	0.8	0.1	0.001	0.64	0.34	0.2	0.0954	0.61	0.1293	0.1293	0.1254	0.00	0.00	0.00	0.1366	0.1254	0.00	0.00	0.1355	0.1254	0.00	0.1355
Scenario 9	0.005	0.8	0.1	0.001	0.84	0.14	0.2	0.0813	0.01	0.0940	0.0940	0.1050	0.00	0.00	0.00	0.1012	0.1053	0.00	0.00	0.1010	0.1053	0.00	0.1010
Scenario 10	0.005	0.8	0.1	0.001	0.04	0.94	0	0.0764	0.00	0.0819	0.0819	0.0973	0.00	0.00	0.00	0.0888	0.0968	0.00	0.00	0.0890	0.0968	0.00	0.0890
Scenario 11	0.005	0.8	0.1	0.001	0.04	0.94	0.1	0.0758	0.00	0.0801	0.0801	0.0972	0.00	0.00	0.00	0.0888	0.0969	0.00	0.00	0.0881	0.0969	0.00	0.0881
Scenario 12	0.005	0.8	0.1	0.001	0.04	0.94	0.2	0.0757	0.06	0.0793	0.0793	0.0969	0.00	0.00	0.00	0.0884	0.0975	0.00	0.00	0.0875	0.0975	0.00	0.0875
Scenario 13	0.005	0.8	0.1	0.001	0.04	0.94	0.3	0.0752	0.07	0.0785	0.0785	0.0969	0.00	0.00	0.00	0.0891	0.0973	0.00	0.00	0.0882	0.0973	0.00	0.0882
Scenario 14	0.005	0.8	0.1	0.001	0.04	0.94	0.4	0.0749	0.12	0.0779	0.0779	0.0965	0.00	0.00	0.00	0.0895	0.0982	0.00	0.00	0.0881	0.0982	0.00	0.0881

Note: The table presents the different simulated scenarios that were assessed to study the robustness of the results.  $\phi_0$ ,  $\phi_1$ , and  $\theta$  are the parameters of the ARMA process;  $\Omega$ ,  $\alpha$ , and  $\beta$  are the coefficients of the GARCH process; and dropout is the dropout rate that is applied during the ML process. RNN is a simple plain vanilla recurrent neural net with one layer and 50 neurons, LSTM is a long short-term memory network with one layer and 50 neurons, and GRU is a gated RNN with one layer and 50 neurons. All models are trained with batch size of 256 days and 1000 epochs. RMSE IS and RMSE OOS are the root mean squared errors between the simulated and predicted returns. The  $p$ -values of the Ljung-Box (LB) test statistic indicate the existence of autocorrelation of the residuals with  $H_0$ . Residuals do not exhibit autocorrelation. The  $p$ -values of the Diebold-Mariano (DM) test statistic indicate the statistical difference between the predicted values of network and a plain vanilla RNN with 50 neurons with  $H_0$ . The generated time series are statistically similar.

TABLE 4 | Empirical study.

	2013-2018		2014-2019		2015-2020		2016-2021		2017-2022	
	In-sample	Out-of-sample	In-sample	Out-of-sample	In-sample	Out-of-sample	In-sample	Out-of-sample	In-sample	Out-of-sample
	2013-2017	2018	2014-2018	2019	2015-2019	2020	2016-2020	2021	2017-2021	2022
$\mu$	0.0004	-0.0002	0.0002	0.0012	0.0004	0.0006	0.0005	0.001	0.0006	-0.0008
$\sigma$	0.0076	0.011	0.0086	0.0085	0.0082	0.0216	0.0129	0.0082	0.0133	0.0151
Median	0.0004	0.0005	0.0003	0.0011	0.0006	0.0024	0.0008	0.0013	0.0012	-0.0015
Min	-0.0402	-0.0418	-0.0418	-0.0302	-0.0418	-0.1277	-0.1277	-0.026	-0.1277	-0.0442
Max	0.0383	0.0484	0.0484	0.0484	0.0484	0.0897	0.0897	0.0235	0.0897	0.054
Range	0.0785	0.0902	0.0902	0.0786	0.0902	0.2173	0.2173	0.0495	0.2173	0.0982
Skewness	-0.4056	-0.4048	-0.4588	-0.0109	-0.6051	-0.8684	-1.1522	-0.3815	-1.0623	-0.0176
Kurtosis	3.1159	2.8838	3.8092	5.5112	4.2484	8.6304	21.2816	0.7234	18.1921	0.3732
N	1026	258	1026	258	1026	258	1026	258	1026	258

Note: The table presents the descriptive statistics of the assessed data set. The data set is divided into five subsamples, and each subsample is divided into a training and validation data set (in-sample) and a test data set (out-of-sample).  $N$  is the absolute amount of observations,  $\mu$  the mean,  $\sigma$  the standard deviation, and min and max are the empirical minimum and maximum.

and training epochs are studied each year. As a result, for each combination of batch size and epochs, we assess 5120 in-sample and 1280 out-of-sample forecasts. Thus, for each network architecture, as described in Table 1, we study 409,600 in-sample and 102,400 out-of-sample forecasts and analyze the respective residuals of each prediction. As a benchmark, we also estimate an ARMA ( $p, q$ ) approach and assess if a better in-sample fit in combination with a higher out-of-sample forecasting precision can be achieved.

However, similar to the results of our simulation exercise, RNNs perform better than LSTM and GRU approaches in this exercise. For the full data set, both LSTM and GRU do not provide a higher in-sample precision and therefore do not outperform the applied ARMA-benchmark. On the other hand, there are many training scenarios in which plain vanilla RNNs with 5, 25, 50, and 100 neurons outperform the ARMA approach in both in-sample and out-of-sample accuracy. Specifically, it is a plain vanilla RNN with 50 neurons trained with larger batch sizes equal to 128 and 256 and 400 or more epochs which describes the best performing approach that statistically outperforms the in-sample RMSE of estimated ARMA-model and also provides a lower out-of-sample RMSE. In order to gain further details into the performance of black-box ML approaches, we discuss selected ML frameworks in more detail and focus on the subsamples to provide an annual assessment of in-sample and out-of-sample performance.

Table 5 summarizes the performance of the ARMA GARCH approach and selected ML architectures for each year. Interestingly, the performance of plain vanilla LSTM and GRU architectures with one layer is comparable and the optimal architecture varies over time. That is, in 2018, 2019, and 2022, one layer with 25 or 50 neurons and batch sizes equal to 128 and 256 describes an adequate choice for both LSTM and GRU. In 2020 and 2021, there is no configuration which outperforms the econometric ARMA Benchmark. Specifically, the in-sample fit provided by both approaches is less precise than ARMA. The performance of plain vanilla RNN with 50 neurons and batch sizes equal to 128 and 256 provide robust results over time. In 2018, 2019, 2020, and 2022, this architecture provides superior in-sample and out-of-sample and outperforms both ARMA as well as LSTM and GRU. In 2021, similar to GRU and LSTM, RNN is not able to outperform the in-sample performance of ARMA model.<sup>5</sup> As described in Table 4, period 2020 is impacted by COVID-related market turmoil,  $\sigma$  increased from 0.0085 in 2019 to 0.0216 in 2020 and indicates that ML approaches are less precise when turmoil market conditions describe the latest information of the training set and the out-of-sample period is described by decreased volatility regime with  $\sigma = 0.0082$ . This finding is also reported in Bhandari et al. (2022); however, RNNs are characterized by higher precision than LSTM and GRU and the in-sample precision, which indicates that RNN adjusts faster to new market conditions. Furthermore, for the subsample period that ranges from 2016 to 2021, accuracy differs only marginally in comparison to ARMA approach.

In addition to in-sample and out-of-sample RMSE, we study the residuals of the estimated returns and find that in contrast to the simulation study, LSTM and GRU networks also provide residuals that are not autocorrelated. As daily returns are generally

TABLE 5 | Empirical study.

					In-sample			Out-of-sample
					2013-2017			2018
Period	Layer	Units	Batchsize	Epochs	RMSE	LB	DM	RMSE
2013-2018								
ARMA (0,0)					0.00760	0.38		0.01099
RNN	1	50	128	200	0.00759	0.47	0.43	0.01097
LSTM	1	50	256	400	0.00757	0.40	0.67	0.01088
GRU	1	50	256	300	0.00759	0.52	0.50	0.01094
<b>2014-2019</b>					<b>2014-2018</b>			<b>2019</b>
	Layer	Units	Batchsize	Epochs	RMSE	LB	DM	RMSE
ARMA(0,0)					0.00864	0.29		0.00791
RNN	1	50	256	700	0.00858	0.34	0.44	0.00787
LSTM	1	100	128	100	0.00864	0.48	0.19	0.00775
GRU	1	50	256	100	0.00863	0.47	0.52	0.00781
<b>2015-2020</b>					<b>2015-2019</b>			<b>2020</b>
	Layer	Units	Batchsize	Epochs	RMSE	LB	DM	RMSE
ARMA(2,2)					0.00817	0.71		0.02171
RNN	1	50	256	1000	0.00815	0.09	0.81	0.02038
LSTM	x	x	x	x	x			x
GRU	x	x	x	x	x			x
<b>2016-2021</b>					<b>2016-2020</b>			<b>2021</b>
	Layer	Units	Batchsize	Epochs	RMSE	LB	DM	RMSE
ARMA (0,2)					0.01230	0.28		0.0086
RNN	x	x	x	x	x			x
LSTM	x	x	x	x	x			x
GRU	x	x	x	x	x			x
<b>2017-2022</b>					<b>2017-2021</b>			<b>2022</b>
	Layer	Units	Batchsize	Epochs	RMSE	LB	DM	RMSE
ARMA (0,2)					0.01282	0.00		0.01585
RNN	1	50	256	100	0.01228	0.00	0.00	0.01409
LSTM	1	50	256	1000	0.01280	0.00	0.01	0.01408
GRU	1	50	256	1000	0.01270	0.00	0.00	0.01524

Note: The table presents the root mean squared error (RMSE) of the in-sample and out-of-sample prediction accuracy of optimal ARMA approach and the best performing simple recurrent neural nets (RNN), long short-term memory (LSTM), and gated recurrent unit (GRU) networks with 5, 25, 50, 75, or 100 neurons and one, two, three, or four layers. For each subperiod, the optimal combination of neurons, batch size, and number of epochs is presented. The  $p$ -values of the Ljung-Box (LB) test statistic (LB) indicate the existence of autocorrelation of the residuals with  $H_0$ ; residuals do not exhibit autocorrelation. The  $p$ -values of the Diebold-Mariano (DM) test statistic (DM) indicate the statistical difference between the predicted values of network and the ARMA-benchmark with  $H_0$ : Two generated time series are statistically similar.

described by weaker autocorrelation than in the applied simulation study, the results indicate that both GRU and LSTM networks provide in-sample residuals that are characterized by similar autocorrelation structure like the underlying training data. Furthermore, similar to the simulation study, we find that additional layers do not result in more precise forecasts, and therefore, we find that for the given daily return framework, plain vanilla RNN describes a solid alternative to econometric

ARMA approach, especially when it comes to daily return assessment with realistic sample sizes.

On this background, we can confirm the result of Bhandari et al. (2022); a single layer LSTM architecture shows better performance than multilayer LSTM. However, for daily financial returns, we find that 50 hidden neurons are sufficient instead of 150 as reported for stock market prices. Also, our results

provide evidence that plain vanilla RNNs provide a higher in-sample and out-of-sample forecasting precision than more complex LSTM and GRU architectures. Due to the fact that trained RNNs capture linear autocorrelation structure, RNNs describe an adequate substitute for ARMA approaches applied to financial time series.

Having in mind that applied portfolio management takes into account expected return and volatilities of financial assets, our findings indicate how artificial intelligence, that is, ML, can be introduced to model expected returns. Further, we demonstrate that plain vanilla RNNs work well with limited training data and that properties of ARMA processes can also be identified via RNNs. Also, our results provide evidence that complex LSTM and GRU networks are outperformed by simple RNNs.

The empirical results add to Yadav et al. (2020) and Bhandari et al. (2022); we also find that single layer networks outperform multilayer networks; however, our results provide evidence that plain vanilla RNNs provide higher precision than LSTM frameworks. Also, we add to the findings of Siami-Namini et al. (2018) who demonstrate that LSTM networks outperform ARMA approaches for monthly returns of Asian and American indices. When the models get trained more than one epoch, we cannot confirm the superior performance of LSTMs.

## 5.1 | Robustness

In order to assess the robustness of our results, we also study the accuracy of the identified RNN, LSTM, and GRU architectures for daily return series of different indices and stocks for the same subperiods. Specifically, we have repeated the presented exercise with major world indices such as NASDAQ-100, S&P 500, Hang Seng Index, FTSE100, DAX 30, Russel 2000, and CAC 40. Also, we have investigated major stocks such as Apple, Microsoft, Amazon, Nvidia, Alphabet Class A, Berkshire Hathaway, United Health Groups, and Exxon Mobil. As a result, the identified plain vanilla RNN with 50 neurons does not get outperformed by the discussed LSTM and GRU architectures. Additionally, we have assessed log returns and heteroscedasticity robust standard errors for the DM-test; our results remain robust.

## 6 | Conclusion

We provide an extensive comparison between econometric time series models and ML techniques in the context of univariate time series analysis. Based on daily financial return series, we study both simulated and empirical data sets and assess forecasting accuracy and residuals of competing approaches.

Taking ARMA as a starting point, this study presents a thorough assessment of hyperparameter tuning for competing ML approaches. In total, we study 80 different training scenarios for competing network architectures and more than 1 million simulated and 512,000 empirical daily forecasts. We find that a RNN with 50 neurons captures generated time-varying autocorrelation structure and therefore describes a solid choice for daily return series. Furthermore, we provide evidence that additional

complexity in terms of architecture and depth does not result in higher forecasting precision for both daily in-sample and out-of-sample predictions.

Moreover, both LSTM and GRU architectures are not able to capture asset specific autocorrelation structure of daily returns, whereas a plain vanilla RNN results in uncorrelated residuals and also outperforms the precision of econometric ARMA approaches. Hence, due to the less restrictive assumptions of RNN, our results provide empirical evidence that RNN with 50 neurons describes a promising alternative to econometric approaches applied to economic daily forecasting exercises based on realistic sample size.

Consequently, our study serves as a reference for daily forecasting studies. An in-depth assessment of RNNs to assess conditional GARCH-variance describes a logical next step to merge the two strings of literature dealing with econometric time series analysis and ML. Also, an application to applied risk measurement, such as Value-at-Risk, describes a promising area for further research.

---

### Acknowledgements

Open Access funding enabled and organized by Projekt DEAL.

### Data Availability Statement

The data that support the findings of this study are available in Yahoo Finance at <https://finance.yahoo.com/>. These data were derived from the following resources available in the public domain: - Yahoo Finance, <https://finance.yahoo.com/quote/%5EGSPC/history/>.

### Endnotes

- <sup>1</sup> Recent literature includes the following: Krauss et al. (2017), Fischer and Krauss (2018), Graves (2014), Giles et al. (2001), Gu et al. (2020), Xiong et al. (2015).
- <sup>2</sup> Please note that we also assess different ARMA parameterizations; the results are presented in Section 4.3.
- <sup>3</sup> Please note that a detailed list of results is available upon request to the authors.
- <sup>4</sup> Please note that a detailed list of results is available upon request to the authors.
- <sup>5</sup> This finding is also valid; if we reduce the dropout rate from 25% to 10%, results are available upon request.

### References

- Avramov, D., S. Cheng, and L. Metzker. 2022. "Machine Learning vs. Economic Restrictions: Evidence From Stock Return Predictability." *Management Science* 69, no. 5: 2587–2619.
- Bengio, Y., P. Simard, and P. Frasconi. 1994. "Learning Long-Term Dependencies With Gradient Descent Is Difficult." *IEEE Transactions on Neural Networks* 5, no. 2: 157–166.
- Berger, T., and R. Gencay. 2018. "Improving Daily Value-at-Risk Forecasts: The Relevance of short-Run Volatility for Regulatory Quality Assessment." *Journal of Economic Dynamics and Control* 92, no. 1: 30–46.
- Bhandari, H. N., B. Rimal, N. R. Pokhrel, R. Rimal, K. R. Dahal, and R. K. C. Khatri. 2022. "Predicting Stock Market Index Using LSTM." *Machine Learning with Applications* 9, no. 1: 100320.

- Bollerslev, T. 1986. "Generalized Autoregressive Conditional Heteroskedasticity." *Journal of Econometrics* 31, no. 1: 307–327.
- Bucci, A. 2020. "Realized Volatility Forecasting With Neural Networks." *Journal of Financial Econometrics* 18, no. 3: 502–531.
- Christiansen, C., M. Schmeling, and A. Schrimpf. 2012. "A Comprehensive Look at Financial Volatility Prediction by Economic Variables." *Journal of Applied Econometrics* 27, no. 1: 956–977.
- Cochrane, J. H. 2011. "Presidential Address: Discount Rate." *Journal of Finance* 66, no. 4: 1047–1108.
- Diebold, F. X. 2015. "Comparing Predictive Accuracy, Twenty Years Later: A Personal Perspective on the Use and Abuse of Diebold-Mariano Tests." *Journal of Business & Economic Statistics* 33, no. 1: 1.
- Diebold, F. X., and R. S. Mariano. 1995. "Comparing Predictive Accuracy." *Journal of Business and Economic Statistics* 13, no. 3: 253–263.
- Engle, R.F., Focardi, S.M., Fabozzi F.J.. 2008. ARCHGARCH Models in Applied Financial Econometrics. In *Chapter in Handbook Series in Finance*, by F. J. Fabozzi, John Wiley & Sons.
- Fernandes, M., M. C. Medeiros, and M. Scharth. 2014. "Modeling and Predicting the CBOE Market Volatility Index." *Journal of Banking & Finance* 40, no. 1: 1–10.
- Fischer, T., and C. Krauss. 2018. "Deep Learning With Long Short-Term Memory Networks for Financial Market Prediction." *European Journal of Operational Research* 270, no. 2: 654–669.
- Fjellström, C. 2022. "Long Short-Term Memory Neural Network for Financial Time Series." *Publications from Uppsala University* 3496–3504. <https://doi.org/10.1109/BigData55660.2022.10020784>.
- Gal, Y., and Z. Ghahramani. 2016. "A theoretically Grounded Application of Dropout in Recurrent Neural Networks." *Advances in Neural Information Processing Systems* 1, no. 1: 1019–1027.
- Giles, C. L., S. Lawrence, and A. C. Tsoi. 2001. "Noisy Time Series Prediction Using Recurrent Neural Networks and Grammatical Inference." *Machine Learning* 44, no. 1: 161–183.
- Goodfellow, I., Y. Bengio, and A. Courville. 2017. *Deep learning*. MIT Press.
- Granger, C. W. 1993. "Strategies for Modelling Nonlinear Time-Series Relationships." *Economic Record* 69, no. 3: 233–238.
- Graves, A., "Generating Sequences With Recurrent Neural Networks." (2014) arXiv preprint, arXiv:1308.0850.
- Gu, S., B. Kelly, and D. Xiu. 2020. "Empirical Asset Pricing via Machine Learning." *Review of Financial Studies* 33, no. 5: 2223–2273.
- Halbleib, R., and W. Pohlmeier. 2012. "Improving the Value at Risk Forecasts: Theory and Evidence From Financial Crisis." *Journal of Economic Dynamics and Control* 36: 1212–1228.
- Hochreiter, S., and J. Schmidhuber. 1997. "Long Short-Term Memory." *Neural Computation* 9, no. 8: 1735–1780.
- Jorion, P. 2007. *Value at Risk. The New Benchmark for Controlling Derivatives Risk*. McGraw.
- Kraus, M., S. Feuerriegel, and A. Oztekin. 2020. "Deep Learning in Business Analytics and Operations Research: Models, Applications and Managerial Implications." *European Journal of Operational Research* 281, no. 1: 628–641.
- Krauss, C., X. A. Do, and N. Huck. 2017. "Deep neural Networks, Gradient-Boosted Trees, Random Forests: Statistical Arbitrage on the S&P 500." *European Journal of Operational Research* 259, no. 1: 689–702.
- Längkvist, M., L. Karlsson, and A. Loutfi. 2014. "A Review of Unsupervised Feature Learning and Deep Learning for Time-Series Modeling." *Pattern Recognition Letters* 42, no. 1: 11–24.
- LeCun, Y., Y. Bengio, and G. Hinton. 2015. "Deep Learning." *Nature* 521, no. 1: 436–444.
- Makridakis, S., and M. Hibon. 2000. "The M3-Competition: Results, Conclusions and Implications." *International Journal of Forecasting* 16, no. 1: 451–476.
- Makridakis, S., E. Spiliotis, and V. Assimakopoulos. 2018. "Statistical and Machine Learning Forecasting Methods: Concerns and Ways Forward." *PLoS ONE* 13, no. 3: e0194889.
- Paye, B. S. 2012. "Déjà vol: Predictive Regressions for Aggregate Stock Market Volatility Using Macroeconomic Variables." *Journal of Financial Economics* 106, no. 1: 527–546.
- Russell, S. J., P. Norvig, and E. Davis. 2010. *Artificial Intelligence: A Modern Approach*. 3rd ed. Upper Saddle River, Prentice Hall.
- Shi, J., M. Jain, and G. Narasimhan. 2022. "Time Series Forecasting Using Various Deep Learning Models." *International Journal of Computer and Systems Engineering* 16, no. 6: 224–232.
- Siami-Namini, S., N. Tavakoli, A. S. Namin. 2018. "A Comparison of ARIMA and LSTM in Forecasting Time Series". In *17th IEEE International Conference on Machine Learning and Applications*, vol 1(1), 1394–1401.
- Sirignano, J., and R. Cont. 2019. "Universal Features of Price Formation in Financial Markets: Perspectives From Deep Learning." *Quantitative Finance* 19, no. 9: 1449–1459.
- Verbeek, M. 2017. *A Guide to Modern Econometrics*, 5th ed. Wiley.
- Vortelinos, D. I. 2017. "Forecasting Realized Volatility: HAR Against Principal Components Combining, Neural Networks and GARCH." *Research in International Business and Finance* 39, no. 1: 824–839.
- Xiong, R., E. P. Nichols, Y. Shen. 2015. "Deep Learning Stock Volatility With Google Domestic Trends." working paper, arXiv preprint arXiv:1512.04916.
- Yadav, A., C. K. Jha, and A. Sharan. 2020. "Optimizing LSTM for Time Series Prediction in Indian Stock Market." *Procedia Computer Science* 167, no. 1: 2091–2100.