Deep Learning for Equity Time Series Prediction

Miquel Noguer Alonso [*1], Gilberto Batres-Estrada [†2], and Aymeric Moulin [‡3]

[1] *Artificial Intelligence in Finance Intitute, NYU Courant. New York. USA* [2] *Trell Technologies, Stockholm, Sweden.* [3] *J.P. Morgan, New York. USA.*

ABSTRACT

We examine the performance of Deep Learning methods applied to equity financial time series. Predicting equity time series is a crucial topic in Finance. To form equity portfolios and do asset allocation, we need to predict returns, compute their risk, and optimize market impact.

One of the modeling benefits of Deep Learning architectures is the ability to model non-linear highly dimensional problems. The lack of transparency and a rigorous mathematical theory could be considered less positive sides. The fact that most progress in Deep Learning has been made by trial and error is also cumbersome. Equity financial time series is a challenging domain with some stylized facts: weak stationarity, fat tails in return distributions, small data sets compared to other areas of Artificial Intelligence (AI), slow decay of autocorrelation in returns, and volatility clustering, to name the most important ones. We perform a comparative study between Long Short Term Memory Networks (LSTM), Recurrent Neural Networks (RNN), Deep Feed Forward Neural Networks (DNN), and Gated Recurrent Unit Networks (GRU). We perform two types of studies. The first focused on a univariate test, and the second a multivariate test.

Our tests show that the LSTM performs the best compared to other Deep Learning and classical machine learning models. In terms of performance metrics, the LSTM is better than the baseline model. We also show that the predictions are better than chance. There is enough evidence that RNN and LSTM can deal with stationary time series and learn the data generating process. Nevertheless, predicting equity non-stationary time series, with market developments like the one caused by the Covid-19 pandemic in 2020, is challenging.

Keywords: Deep Learning, Neural Networks, Recurrent Neural Networks, Long Short Term Memory Networks, Back-propagation through time, AI, Finance, stock prediction, time series.

[*]miquel.noguer@aifinanceinstitute.com

[†]gilberto@trell.se

[‡]aymeric.moulin@columbia.edu

I. Introduction

The stock price is assumed to contain all information available for that stock in the financial markets. Price data capture the complexity of feedback from the different agents that participate in the market.

Complex system is a broad term encompassing a research approach to problems in diverse disciplines, including statistical physics, information theory, non-linear dynamics, anthropology, computer science, meteorology, sociology, economics, psychology, and biology. Financial markets rely on strong mathematical assumptions necessary for forecasting and understand the stochastic evolution of financial asset prices and returns. Such assumptions are that stochastic processes and random variables are independent, identically distributed, and stationary. Additionally, it is assumed that the sample size is big enough to estimate these stochastic processes. Unless these conditions are met, it is difficult or impossible to have a guaranteed generalization constrained by the irreducible error.

Machine learning models have gained momentum in finance applications for the last five years following the tremendous success in areas like image recognition and natural language processing. These models have proven to be very useful to model unstructured data. In addition to that, machine learning models can model non-linearity in classification and regression problems and discover hidden structures in Supervised Learning. In Supervised Learning, there is an input data set as well as its corresponding labels. Combinations of weak learners like XGBoost and Adaboost are widely popular for Supervised Learning tasks.

Unsupervised Learning is a branch of machine learning used to draw inferences from datasets consisting of input data without labeled responses. Principal Component Analysis is an example of Unsupervised Learning. The challenges of using machine learning in Finance are many. One issue which is hard to accept is model interpretability, or that machine learning models tend to overfit. Several good references [1], [2] and [3] explore in-depth machine learning algorithms in Finance.

Our paper is ordered as follows. In section (I), we motivate this work and the use of Deep Learning in Finance. In SectionIII, we review Deep Learning in the finance literature. We continue with section II, which introduces Deep Learning in general. Section V covers the financial problem we try to solve with LSTM, the data used, and methods. In the same section, we present the results. Section VI concludes.

II. Deep Learning

Deep Learning is a popular area of research in machine learning and Artificial Intelligence (AI). One reason is that it has achieved significant success in computer vision, natural language processing, machine translation, and speech recognition. Another reason is that it is now possible to build applications based on Deep Learning, [4].

Some examples of everyday applications based on Deep Learning are recommendation systems, voice assistants, and search engine technology based on computer vision. Deep Learning enjoys success, thanks to the amount of data now available, known as big data. The introduction of the Graphical Processing Unit (GPU) allows researchers and engineers to perform computation much faster than ever before, which also allows faster experiment iteration. Another approach is to move computations to clusters of computers in local networks or the cloud.

The term Deep Learning refers to training deep neural networks. There are many different types of architectures and different methods for training these models. The most common network is the Feed Forward Neural Network used for data assumed to be independent and identically distributed (i.i.d). On the other hand, recurrent neural networks are more suitable for sequential data, such as time series, speech data, and natural language data, [17]. In Deep Learning, the main task is to learn or approximate some function f that maps inputs $\boldsymbol{x}$ to outputs $\boldsymbol{y}$. Deep Learning tries to solve the following learning problem $\hat{\boldsymbol{y}} = f(\theta; \boldsymbol{x})$, where $\boldsymbol{x}$ is the input to the neural network, $\hat{\boldsymbol{y}}$ is its output and θ is a set of parameters that give the best fit of f. In regression, $\hat{\boldsymbol{y}}$ would represent real numbers, whereas, in classification, it would represent probabilities assigned to each class in a discrete set of values.

Recurrent Neural Networks (RNNs) capture sequential order and are suitable for processing sequential data. RNNs are powerful models due to their ability to scale to much longer sequences than it would be possible for regular neural networks. They suffer from two serious problems: the first has to do with vanishing gradients and the second with exploding gradients [5, 6, 7, 22], both solved by the Long Short-Term Memory network (LSTM), [7]. In recent years, LSTMs have solved many speech recognition and machine translation problems, where the goal is often to match an input series to an output series. The Long Short-Term Memory network can solve both classification and regression problems.

For comparison, we perform experiments with Gated recurrent units (GRU's). GRUs are similar to LSTMs and were introduced in 2014 by [8]. GRU consists of an update gate and a reset gate. Its performance is similar to that of LSTMs on many tasks. In [9], the authors show that the LSTM is "strictly stronger" than the GRU as it can efficiently perform unbounded counting, which seems not to be the case for GRU. That is why the GRU fails on simple natural language processing (NLP) tasks.

New architectures dealing with sequences based on the concept of attention have emerged. The attention mechanism was developed to memorize long source sentences in NLP tasks. Rather than using one context vector out of the encoder's last hidden state, attention creates shortcuts between the context vector and the entire source input. The weights of these shortcut connections can be customized for each output element as the context vector can access the entire input sequence. The alignment between the source and target is learned and controlled by the context vector. The paper "Attention is All you Need" [10] proposed a "transformer" model using self-attention mechanisms without using sequence-aligned recurrent architectures.

III. Deep Learning in Finance Literature

Researchers and practitioners have been working for decades on methods to improve forecasting accuracy in Finance. For many years there was little research on Finance and neural networks, especially using recurrent neural networks.

- In [11], we explored the performance of several recurrent neural network architectures. We also provide a review of time series in Finance as well as recurrent neural networks.
- Dixon introduces Exponentially Smoothed Recurrent Neural Networks [12]. He also gives a well and extensive coverage of the topic in [2].
- In this paper [13], the authors introduce Deep Momentum Networks – a hybrid approach that injects Deep Learning based trading rules into the volatility scaling framework of time-series momentum.
- [14] makes an exhaustive study comparing many machine learning algorithms and shows that LSTMs outperform the other models in the study.
- In [15], we can see a thorough analysis of our current understanding of the mathematical properties of deep neural networks.
- The paper [16] study the construction of portfolios and focus on ten stocks to trade. They achieve good results in constructing portfolios that exhibit a consistent risk-return profile at various thresholds levels. Their LSTM has a hidden layer with 100 hidden units.

Our paper contributes to a better understanding of several Deep Learning architectures' performance in the univariate and multivariate equities prediction domain.

IV. Stock Return Estimations Using Deep Learning

We use Deep Learning techniques to predict daily future stock returns. We focus on predicting daily returns for a group of liquid stocks from the S&P 500.

We conducted two experiments. The first one consists of a single stock approach where only the historical data of stock s_t is used to estimate future returns of stock s_{t+1}. In the second experiment, we estimate future returns for stocks in a basket $\mathcal{S}$ by using historical data of the stocks in the

basket $\mathcal{S}$ and other extra additional assets $\mathcal{E}$ in order to build a strategy for managing an equity portfolio based on basket $\mathcal{S}$.

A. Definitions

In the following list, we summarize some financial quantities used in our experiments.

1. Stock returns are computed with the formula: $R^i_{t+1} = \frac{P^i_{t+1} - P^i_t}{P^i_t}$, where P^i_{t+1} is the price of asset i at time t.

2. Deep Learning is used for estimating a function, G_θ, which depends on stock returns and where θ is a vector of parameters. We divide the estimation according to the two sets of experiments we performed:

   (a) For the univariate case (experiment 1): $\hat{R}^i_{t+1} = G_{\theta_i}(R^i_t, R^i_{t-1} \ldots, R^i_{t-k+1})$
   (b) For the multivariate case (experiment 2): $\hat{R}_{t+1} = G_\theta(R_t, R_{t-1} \ldots, R_{t-k+1}; E_t, E_{t-1} \ldots, E_{t-k+1})$, with $R_t = (R^i_t)_{i \in \mathcal{S}}$ vector of returns of stocks in $\mathcal{S}$ at time t and $E_t = (E^i_t)_{i \in \mathcal{E}}$ vector of returns of extra assets in $\mathcal{E}$ at time t.

3. Look-back: look-back, k in the index in the above expressions, is the length of historical data counted as the number of days. The number of time steps (days here) is used by the model to estimate future returns.

B. Model Evaluation

In what follows, we present the error metrics and the loss functions used in our experiments.

1. For the univariate (experiment 1) case we employed the Huber loss defined as:

$$L(R_{t+1}, G_\theta) = \begin{cases} \frac{1}{2}\left[R_{t+1} - G_\theta\right]^2 & \text{for } |R_{t+1} - G_\theta| \leq \delta, \\ \delta\left(|R_{t+1} - G_\theta| - \delta/2\right) & \text{otherwise,} \end{cases} \tag{1}$$

   where R_{t+1} is the target and $G_\theta(R^i_t, R^i_{t-1} \ldots, R^i_{t-k+1})$ is the function estimated with Deep Learning techniques, and $\delta = |R_{t+1} - G_\theta(R^i_t, R^i_{t-1} \ldots, R^i_{t-k+1})|$ is the residual. Note that we write G_θ and leave out the argument in order to make the expressions more readable.

2. We use the Mean Squared Error (MSE) both as an error metric and as the loss function in the multivariate regression problem: $MSE = \frac{1}{|Samples|} \sum_{t \in\ Samples} \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} (\hat{R}^i_t - R^i_t)^2$.

3. The Hit-Ratio (HR) is defined as the fraction of times the Deep Learning model predicts the returns going in the same direction as real returns. We use HR to get results in a manner that reminds us of the accuracy metrics employed in classification problems. For our experiments, we define the following hit ratios:

(a) For experiment 1: $HR_i = \frac{1}{|Samples|} \sum_{t \in Samples} \mathbb{1}_{\hat{R}_t^i \cdot R_t^i > 0}$
(b) For experiment 2: $HR = \frac{1}{|Samples|} \sum_{t \in Samples} \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \mathbb{1}_{\hat{R}_t^i \cdot R_t^i > 0}$

V. Experiments

A. Single Stock Predictions

The input to the LSTM was defined in the Financial Model section. The stock returns were then normalized according to the min-max formula:

$$x' = \Delta x \cdot (b - a) + a, \tag{2}$$

where $\Delta x = \frac{x - \min(x)}{\max(x) - \min(x)}$, and a, b is the range (a, b) of the features. It is common to set $a = 0$ and $b = 1$.

For the single stock prediction experiment, our model is trained with the stock returns at time step t as input to predict the return at time step $t + 1$. We train one model for each stock. The data was divided into three portions, 75% train set, 12.5% validation set, and 12.5% test set. We use data for those stocks where data is available from 1980-01-01 to 2020-07-31.

A.1. Training and Hyper Parameter Search

The hyper-parameters of the LSTM were chosen by a method called Hyperband [21]. According to the authors, Hyperband is a pure-exploration non-stochastic infinite-armed bandit algorithm. This algorithm combines adaptive resource allocation and early stopping. According to the Hyperband authors, it can achieve a speed-up of an order of magnitude compared to Bayesian methods on a set of Deep Learning and kernel-based learning problems. We used the Keras Tuner implementation of Hyperband in our experiments. For both LSTM and other Deep Learning models tested in this experiment, we used the same hyper-parameter set as input for Hyperband to search for the optimal set of hyper-parameters. Preliminary experiments suggested that deeper networks gave us similar results as those with only one hidden layer. Also, we noticed that deeper networks with many hidden units were computationally costly. These early results lead us to choose to work with one-hidden layered neural networks.

The number of hidden units tested by Hyperband was in the range of 30 up to 250. For each iteration, ten new hidden units were added to the hidden layer. We experimented with activation functions in the set {ReLu, ELU, SELU}. Dropout [25] was used between the hidden layer and the output layer. The Dropout varied from 0 to 0.4 with a step of 0.1 for each iteration of Hyperband. The learning rate was chosen from the set {1e-2, 1e-3}. For optimizer, we chose RMSProp. As a loss function, we used the Huber loss, which is more robust against outliers. We retrained each of the neural networks used in the experiments with the hyper-parameters chosen by Hyper-

band. Additionally, we used exponential decay for the learning rate during the final retraining after Hyperband. For comparison, we trained a simple Recurrent Neural Network (RNN), a Gated Recurrent Unit (GRU), and a Deep Neural Network (DNN). As a baseline model, we used a dummy model. The dummy model used the daily returns at time step t as predictions for time step $t + 1$. Additionally, a Support Vector Machine was trained to compare Deep Learning models to classical machine learning models.

Table I, shows the results for the experiments done with the LSTM neural network. The results shown in the table are the Hit-Ratio (HR), the MSE and the MAE. The following columns in the set: {Units, Activation, Dropout, Learning-rate} show the hyperparameters found by Hyperband search. For the other models the results are shown in the Appendix in tables A1, A2, A3 and A4. Table A1 shows the results for GRU, Table A2 shows the results for RNN, Table A3 shows the results for DNN and Table A4 shows the results for both the baseline model and Support Vector Machine (SVM).

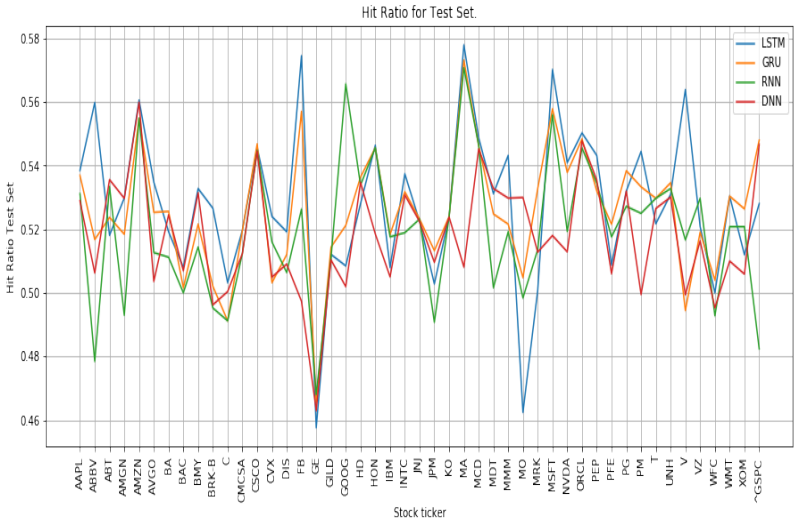

Figure 1. Hit Ratio for test set. In the plot we can see LSTM compared to the other Deep Learning models tested, GRU, RNN and DNN.

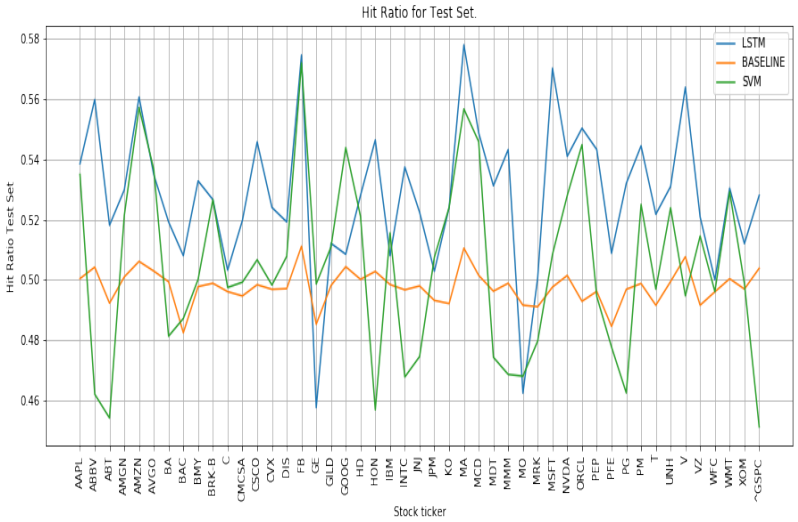

Figure 2. Hit Ratio for test set. In the plot we can see LSTM compared to the baseline model and Support Vector Machine.

To aid in comparing results, we plot the hit-ratio for each model's test set and stock. For clarity, we choose to show the hit-ratio for all the Deep Learning models in Figure 1. In Figure 2, we see the LSTM's hit-ratio result against the baseline model and Support Vector Machine. For the Support Vector Machine (SVM), we set the following values for the parameters: $C = 1$ and its gamma parameter $\gamma = 0.1$. Compared to the baseline model and to the SVM, LSTM performs much better. Compared to other Deep Learning models, it is not that clear. For certain stocks, the Deep Learning models perform very similarly. We show both the Mean Squared Error (MSE) and the Mean Absolute Error (MAE) computed on the test set in the Appendix. In general, we can say that the Deep Learning models have lower values for the error metrics than both the baseline model and SVM.

Both in terms of accuracy (HR) and error metrics (MSE, MAE), the Deep Learning models follow the stock returns' movements. Among the Deep Learning models, only the DNN seems to have a higher error or lower accuracy, suggesting that sequential models seem to predict stock returns' movements better.

Stock	Units	Activation	Dropout	Learning-rate	Hit-Ratio Test Set	MSE Test Set	MAE Test Set
XOM	90	elu	0.2	0.010	0.50	0.000303	0.011165
WMT	170	relu	0.4	0.001	0.53	0.000204	0.008870
WFC	240	selu	0.0	0.001	0.49	0.000420	0.012740
VZ	110	relu	0.1	0.001	0.52	0.000150	0.008539
V	100	relu	0	0.001	0.56	0.000510	0.013884
UNH	180	relu	0.0	0.010	0.53	0.000329	0.011101
T	130	relu	0.0	0.010	0.53	0.000218	0.009562
PG	150	relu	0.0	0.010	0.53	0.000164	0.008031
PFE	140	relu	0.4	0.001	0.51	0.000198	0.009459
PEP	80	relu	0.4	0.001	0.54	0.000171	0.007837
PM	190	elu	0	0.001	0.54	0.000496	0.014209
ORCL	130	relu	0.0	0.010	0.55	0.000285	0.009782
NVDA	120	elu	0.0	0.001	0.54	0.001009	0.022151
MSFT	160	relu	0.0	0.010	0.56	0.000299	0.010959
MRK	30	relu	0.0	0.010	0.52	0.000206	0.009721
MO	230	elu	0.0	0.010	0.53	0.000238	0.010355
MMM	190	selu	0.0	0.001	0.54	0.000236	0.009748
MDT	110	relu	0.4	0.001	0.53	0.000234	0.009770
MCD	220	selu	0.0	0.001	0.55	0.000221	0.008648
MA	190	relu	0	0.01	0.58	0.000611	0.015243
KO	120	relu	0.3	0.010	0.52	0.000153	0.007801
JPM	210	selu	0.0	0.001	0.52	0.000351	0.011632
JNJ	140	relu	0.1	0.001	0.52	0.000160	0.007981
INTC	200	relu	0.0	0.010	0.54	0.000427	0.012986
IBM	160	relu	0.0	0.010	0.51	0.000261	0.010298
HON	190	relu	0.4	0.010	0.55	0.000244	0.009587
HD	110	relu	0.0	0.010	0.53	0.000264	0.009960
GSPC	50	elu	0	0.001	0.53	0.000151	0.007160
GOOG	170	selu	0	0.001	0.51	0.000444	0.013833
GILD	180	relu	0.0	0.001	0.51	0.000316	0.009960
GE	210	relu	0.4	0.010	0.46	0.000592	0.009960
FB	180	relu	0	0.001	0.57	0.000712	0.017606
DIS	100	relu	0.1	0.010	0.51	0.000274	0.010371
CVX	30	selu	0.0	0.010	0.52	0.000424	0.012226
CSCO	50	relu	0.3	0.001	0.54	0.000320	0.011435
CMCSA	160	relu	0.0	0.010	0.53	0.000253	0.010970
C	50	relu	0.4	0.010	0.50	0.000539	0.014319
BMY	110	relu	0.2	0.010	0.53	0.000290	0.011392
BAC	160	relu	0.0	0.010	0.51	0.000470	0.014114
BA	80	relu	0.0	0.010	0.52	0.000756	0.015532
BRK-B	80	relu	0.2	0.001	0.52	0.000255	0.009844
AVGO	40	relu	0	0.001	0.53	0.000927	0.019627
AMZN	180	relu	0.0	0.010	0.56	0.000432	0.014260
AMGN	150	relu	0.0	0.010	0.54	0.000262	0.010848
ABT	40	relu	0.0	0.001	0.53	0.000256	0.010783
ABBV	160	relu	0	0.01	0.56	0.000493	0.014903
AAPL	80	relu	0.1	0.010	0.54	0.000334	0.012059

Table I Single stock predictions: Results for LSTM, the performance is measured as the Hit-Ratio (HR), Mean Squared Error (MSE) and Mean Absolute Error (MAE). The results are computed for the independent test set.

A.2. Results for Single Stock Predictions

We conclude that Deep Learning models are more accurate for the single stock return prediction experiments than SVM. The results suggest that Deep Learning methods learn to find correlations from the noisy time series and that their predictions are not a matter of luck. Especially the model we focus on in this study, LSTM seems to be better than the other Deep Learning methods. There is only one exception: DNNs seem to have more difficulties predicting the direction of the returns' movements and tend to have a higher error, both measured as MSE and MAE. There is no question

that LSTM is much better than the baseline model and SVM. At the same time, we can see that accuracy, as measured by hit-ratio, is not that high as the accuracy in other machine learning tasks such as in computer vision. It is a known fact that predicting financial time series is a real challenging task.

B. Multiple Stock Predictions

In this experiment, the goal is to fit one neural network that outputs a vector predicting the returns of the 50 stocks. Each output is connected to the same hidden layers. The hope is that the hidden layers learn a robust general representation that is predictive for each stock. The model takes as input the time series of the 50 stocks' returns and, additionally, S&P 500, oil, and gold. We will not present the lengthy grid search and trial and error process to justify the architecture type choice in presenting this experiment. Rather, we will focus on the case of a straightforward architecture with only one hidden layer constituted of a single low dimensional LSTM cell.

B.1. Assessing Performance

To assess the quality of the prediction, we use conventional MAE and MSE. We also use the hit-ratio (HR) for computing the ratio for the number of times the model is right in its predictions. We build a straightforward strategy using HR prediction accuracy. The long-short strategy consists of updating a portfolio daily by holding a long position on stocks where we predict a positive return and a short position on stocks where we predict a negative return (or long position and flat position for the long-only strategy).

The data set is a five-year time horizon divided into 75/12.5/12.5% portions representing the training, validation, and test sets, respectively. The model is trained on the training set, hyper-parameter tuned on the validation set, and the final architecture's prediction power is measured on the test set. We track the evolution of performance across the training epochs by evaluating the training set metrics and out of sample on the validation set. For practical reasons, the metrics are computed on each epoch's full validation set but only a random selection of batches from the training set. For each epoch, we only monitor the average return of the strategies. However, once the final architectures are chosen and trained, we visualize the strategies' cumulative return on the different sets. Then we compare them to the benchmark strategy, consisting of holding the basket of stock long on the full-time period.

B.2. Results for Multiple Stock Predictions

LSTM architectures with one hidden layer seem complex enough to learn on the training set but simple enough to avoid overfitting. Through trial and terror with different architectures, we

found it interesting that one hidden layered LSTM performed significantly better than any other small number of cells. Although the architecture seems to be the right choice for 2014-2019, it does not as well for other previous time horizons comparing performance to the buy and hold the basket benchmark. Further work is required to understand what in the market environment makes the model chosen perform poorly on those previous horizons.

Focusing on the time horizon 01/09/2014-01/09/2019, we can make the following inferences. For look-back windows of 15 and 25 days, the MSE computed on the training set, decreases through the epochs, whereas the MSE on the validation set decreases until around epoch 200 before starting to go back up. MAE exhibits similar behavior. Both hit ratio and strategy average return exhibit the highest values around epoch 200. While validation MSE and MAE increase, average returns and hit ratios stay stable and increase slightly. It can be observed in the cumulative return figures that the buy and hold benchmark is beaten by the strategies proposed based on the predictions.

It is also worth looking at the results around the Covid-19 related market crash in 2020 (charts in the Appendix). The network weights are computed before the year 2020, which means that our predictions for the Covid-19 market crash period are entirely out of sample. We can see in the training and validation plots that the model learns market behavior before 2018. We confirm that it outperforms the benchmark on the validation set as well. In the out of sample data from 2020, we can see how the model leads to choices that significantly reduce the draw-down effect in March. It leads to the model beating the benchmark consisting of just holding the basket long on the whole period.

Start	End	Sharpe Train Set	Sharpe Validation Set	Sharpe Test Set
2015-07-31	2020-07-31	3.8 (L)/5.2 (LS)	2.8 (L)/4.2 (LS)	1.7 (L)/2.3 (LS)
2014-09-01	2019-09-01	3.7 (L)/4.6 (LS)	2.0 (L)/2.2 (LS)	0.6 (L)/1.8 (LS)
2013-09-01	2018-09-01	4.2 (L)/5.0 (LS)	4.0 (L)/5.2 (LS)	0.7 (L)/-0.7 (LS)
2012-09-01	2017-09-01	4.7(L)/5.5(LS)	4.1 (L)/5.9 (LS)	2.0 (L)/1.9 (LS)
2011-09-01	2016-09-01	4.3(L)/5.0(LS)	2.1 (L)/3.2 (LS)	1.0 (L)/1.2 (LS)
2010-09-01	2015-09-01	3.8(L)/4.9(LS)	3.1 (L)/3.2 (LS)	1.1 (L)/0.9 (LS)

Table II Sharpe Ratio (SR) for train, validation and test sets. In each of SR-column the (L) stands for Long portfolio (L) an (LS) for Long-Short portfolio.

B.3. Comment about cell type

As mentioned earlier, this section aims not to present a trial and error or grid search over all architectures possible. However, it might be worth mentioning that for the minimalist one cell layer architecture chosen here in this paper, switching the LSTM cell for a GRU cell leads to under-fitting

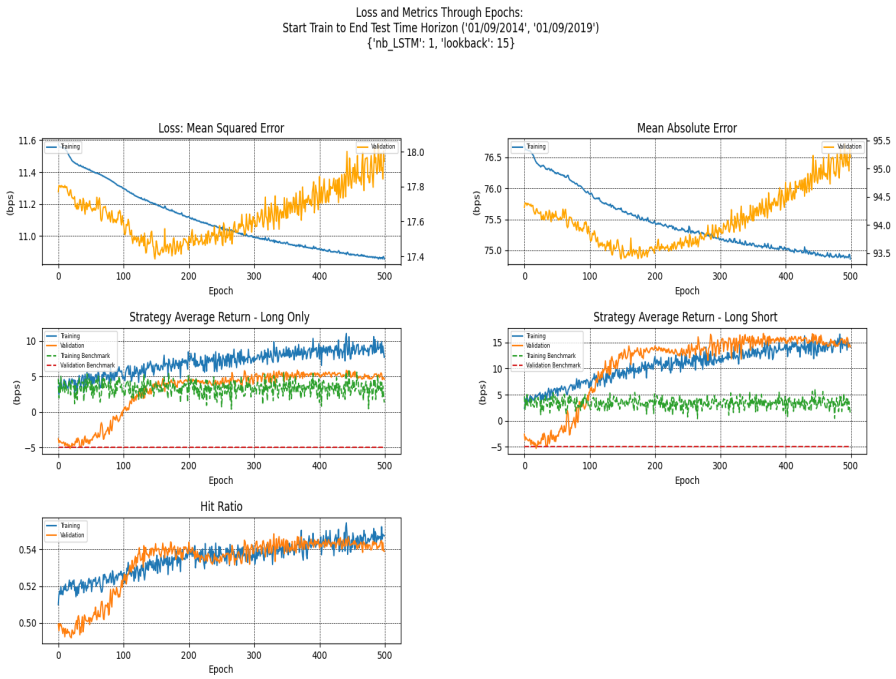

Figure 3. Training and validation metrics for a single LSTM cell as hidden layer with lookback 15 on the start training to end testing horizon 2014-09-01 to 2019-09-01

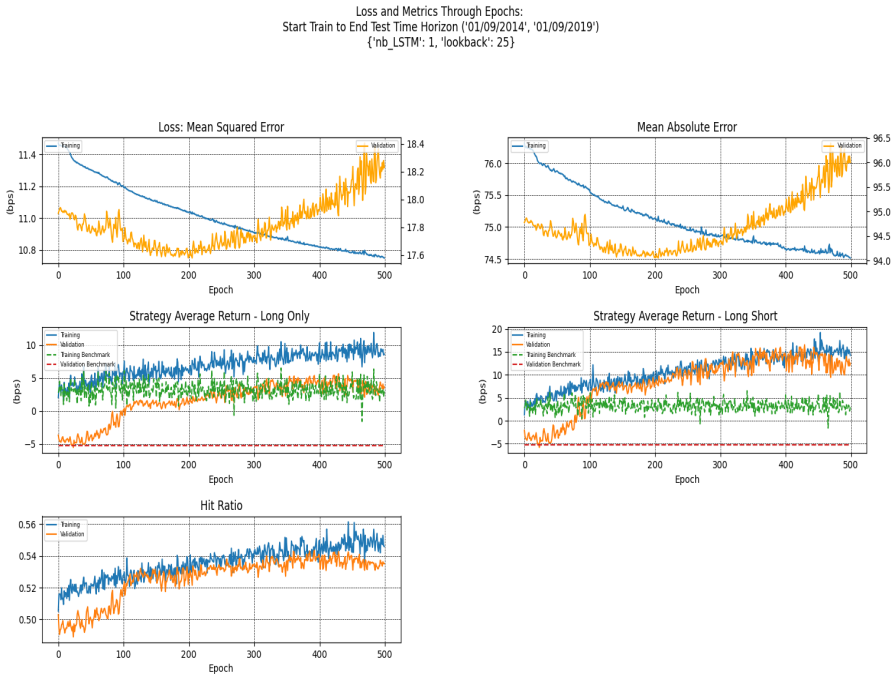

Figure 4. Training and validation metrics for a single LSTM cell as hidden layer with lookback 25 on the start training to end testing horizon 2014-09-01 to 2019-09-01

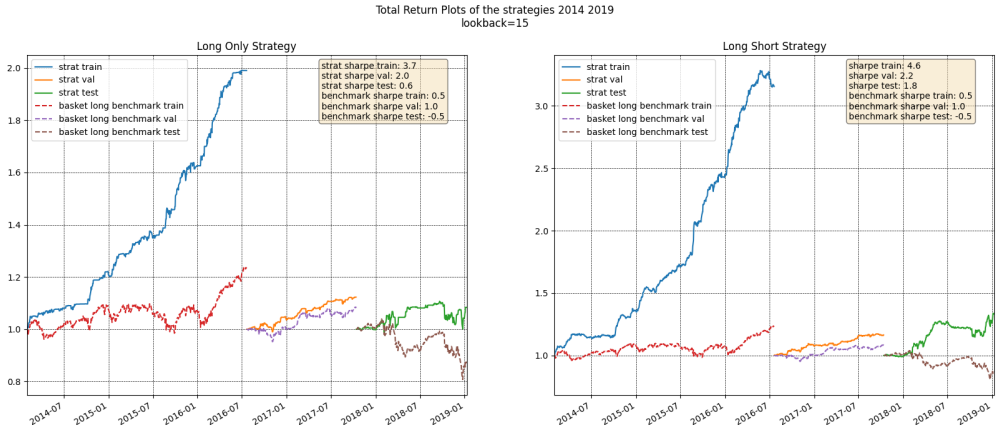

Figure 5. Cumulative return of the strategies compared to the benchmark consisting in holding the basket long for the full period for a single LSTM cell as hidden layer with lookback 15 on the start training to end testing horizon 2014-09-01 to 2019-09-01

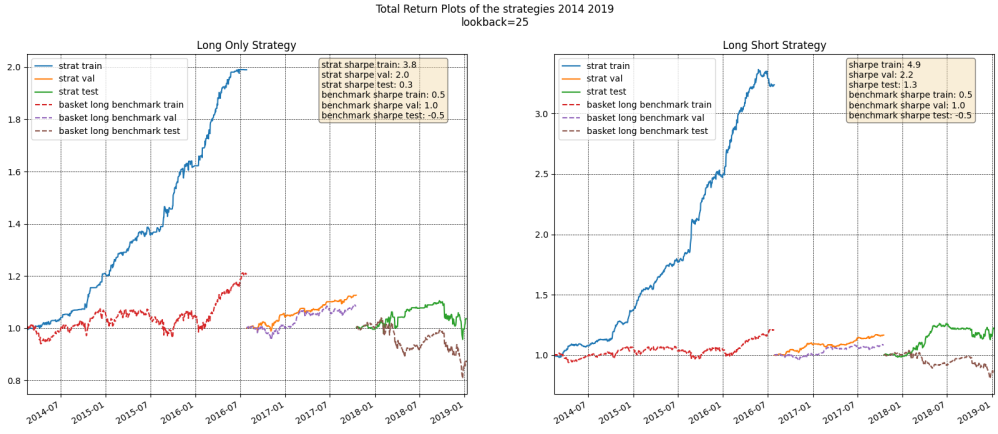

Figure 6. Cumulative return of the strategies compared to the benchmark consisting in holding the basket long for the full period for a single LSTM cell as hidden layer with lookback 25 on the start training to end testing horizon 2014-09-01 to 2019-09-01

and decreases the model and strategy's performance. Similar behavior is observed for RNN cells.

To compensate for the apparent lack of flexibility in the under-fitting model, we tried to add more cells while keeping the number low. Intuitively it would seem that if a single LSTM cell works well and a single GRU cell under-fits a bit, then a few GRU cells should do the job. It did not work. Results were not as good as with a single LSTM. It seems like there is a benefit in using a minimalist model with a single cell.

VI. Conclusions

We examine the performance of Deep Learning methods applied to equity financial time series. The aim has been to predict future returns. Predicting equity time series is an important problem in Finance. To form equity portfolios, we need to predict future returns. Combining risk and market impact with the theory of time series and factor models produces the best prediction possible. Once we have these two objects, we then need to perform asset allocation. Classical econometrics and finance theory has produced fundamental models to model financial time series.

The study of equity financial time series is a challenging domain with some stylized facts: weak stationarity, fat tails, small data sets, slow decay of autocorrelation in returns, and volatility clustering, to name the most important ones. One of the modeling benefits of Deep Learning architectures is modeling non-linear, high dimensional problems. The lack of transparency and a rigorous mathematical theory could be considered as less positive sides. Most of the progress in Deep Learning has been made by trial, and error is also cumbersome.

We performed two case studies, one focused on predicting stock returns. Here we consider the prediction task as a univariate problem. The second study focuses on predicting the returns as a multivariate problem. In the univariate tests, we study the performance of Long Short Term Memory Networks (LSTM) compared to some of the most common machine learning algorithms, such as Support Vector Machines (SVM), Recurrent Neural Networks (RNN), and Gated Recurrent Units (GRU). Our study suggests that LSTM shows a better performance compared to the other tested models. By measuring the hit-ratio and the error metrics such as MSE and MAE, we see that LSTM achieves a better performance. The training, validation, and test periods for the univariate case can be found in the Appendix.

In the multivariate stock data experiment, we performed tests to compare LSTMs to GRUs. The GRU experiments showed worse performance than LTSM. Therefore our analysis focuses on training LSTM networks for different periods. Using a simple architecture can produce good Sharpe ratios on the long and long-short portfolios.

There is enough evidence that RNN and LSTM can deal with stationary time series and learn the data generating process. Nevertheless, predicting equity non-stationary time series, with market

developments like the one caused by the Covid-19 pandemic in 2020, is challenging.

We provide an extensive examination of the performance in different periods to assess investment performance for our portfolios. By comparing the performance on the training set and the test set, we can see that overfitting is small and acceptable. Future research will involve adding exogenous variables to the experiments.

VII. Appendix

A. Single Stock Experiment Additions

This section shows plots for the Mean Squared Error (MSE) and for the Mean Absolute Error (MAE) for the univariate case. Both the MSE and MAE are computed on the test set. We note that both the MSE and MAE are lower for the LSTM than the baseline model and SVM. Comparing the Deep Learning models, we can note that most of them have similar performance. The only exception is the DNN model, which seems to have a higher error for some stocks. We conclude that DNNs have more difficulty predicting the stock returns than the sequential models.

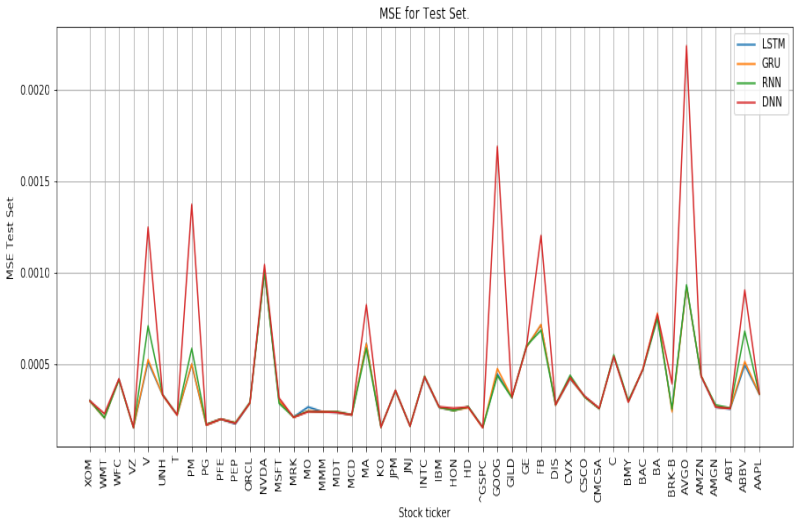

Figure 7. Mean Squared Error (MSE) for test set. In the plot we can see LSTM compared to the other Deep Learning models, GRU, RNN, and DNN.

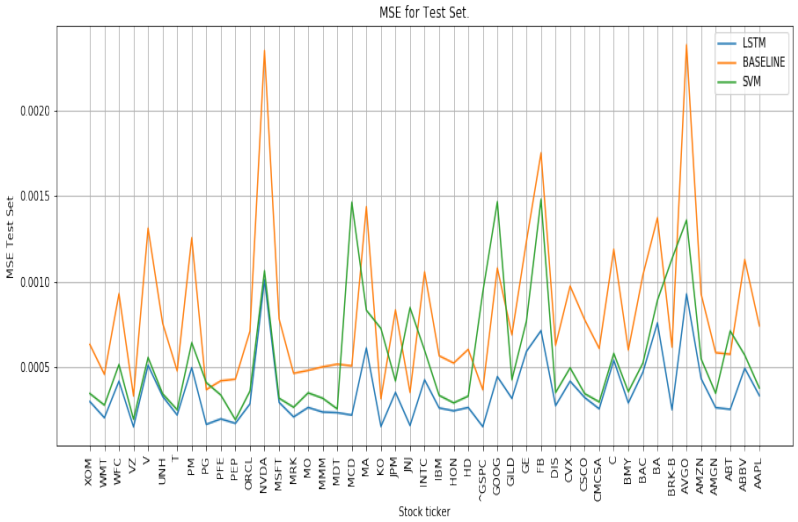

Figure 8. Mean Squared Error (MSE) for test set. In the plot we can see LSTM compared to the baseline model and Support Vector Machine.

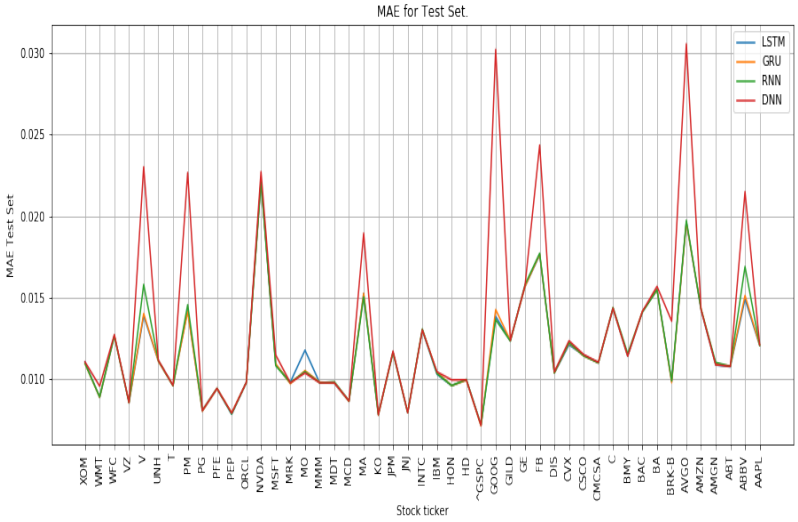

Figure 9. Mean Absolute Error (MAE) for test set. In the plot we can see LSTM compared to the other Deep Learning models, GRU, RNN, and DNN.

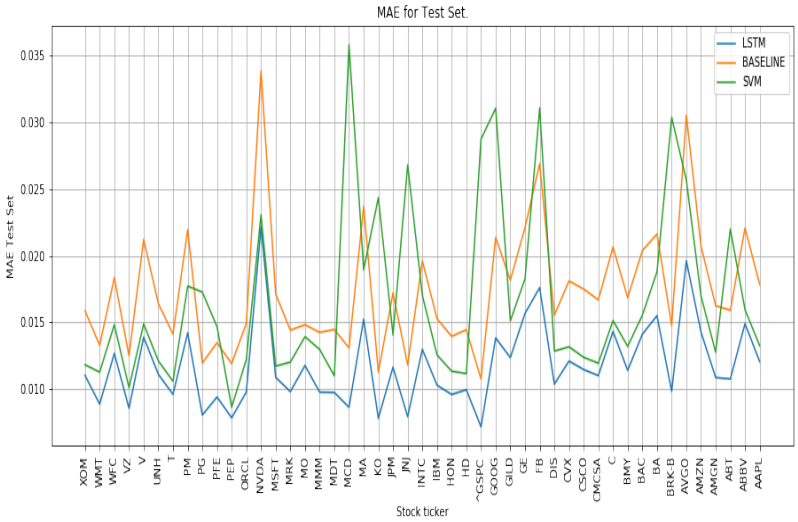

Figure 10. Mean Absolute Error (MAE) for test set. In the plot we can see LSTM compared to the baseline model and Support Vector Machine.

Stock	Units	Activation	Dropout	Learning-rate	Hit Ratio Test Set	MSE Test Set	MAE Test Set
XOM	120	relu	0.2	0.001	0.51	0.000297	0.010954
WMT	40	relu	0.4	0.001	0.53	0.000204	0.008872
WFC	250	elu	0.2	0.001	0.50	0.000420	0.012748
VZ	70	elu	0.1	0.010	0.52	0.000150	0.008543
V	120	elu	0.3	0.001	0.49	0.000523	0.014015
UNH	90	relu	0.3	0.001	0.53	0.000329	0.011116
T	30	relu	0.4	0.001	0.47	0.000228	0.009827
PM	190	relu	0.1	0.001	0.53	0.000499	0.014130
PG	240	relu	0.1	0.001	0.53	0.000165	0.008050
PFE	220	relu	0.3	0.001	0.51	0.000198	0.009441
PEP	70	elu	0.0	0.001	0.54	0.000170	0.007828
ORCL	200	relu	0.2	0.001	0.55	0.000284	0.009770
NVDA	70	relu	0.4	0.010	0.54	0.001014	0.022236
MSFT	70	relu	0.2	0.001	0.55	0.000304	0.011036
MRK	70	elu	0.1	0.010	0.53	0.000206	0.009706
MO	50	selu	0.4	0.001	0.53	0.000238	0.010373
MMM	120	selu	0.3	0.001	0.53	0.000238	0.009779
MDT	210	relu	0.3	0.001	0.53	0.000235	0.009785
MCD	110	relu	0.1	0.010	0.55	0.000219	0.008620
MA	90	relu	0.1	0.001	0.57	0.000610	0.015253
KO	100	relu	0.2	0.001	0.53	0.000152	0.007780
JPM	70	elu	0.3	0.001	0.50	0.000350	0.011637
JNJ	130	relu	0.2	0.001	0.52	0.000159	0.007967
INTC	160	relu	0.4	0.001	0.53	0.000430	0.013008
IBM	50	elu	0.3	0.001	0.52	0.000260	0.010284
HON	250	elu	0.4	0.001	0.55	0.000244	0.009591
HD	240	selu	0.3	0.001	0.53	0.000268	0.009984
GSPC	70	relu	0.2	0.001	0.54	0.000150	0.007122
GOOG	120	elu	0.3	0.001	0.52	0.000474	0.014257
GILD	180	elu	0.3	0.001	0.52	0.000315	0.012338
GE	50	elu	0.4	0.001	0.47	0.000594	0.015749
FB	230	relu	0.2	0.001	0.56	0.000715	0.017652
DIS	180	relu	0.1	0.001	0.51	0.000275	0.010378
CVX	160	relu	0.1	0.001	0.52	0.000417	0.012140
CSCO	30	relu	0.4	0.001	0.55	0.000323	0.011628
CMCSA	240	relu	0.2	0.010	0.52	0.000255	0.011000
C	210	relu	0.3	0.001	0.49	0.000543	0.014350
BMY	200	selu	0.3	0.001	0.51	0.000294	0.011474
BAC	40	selu	0.4	0.010	0.50	0.000473	0.014155
BA	40	elu	0.2	0.001	0.54	0.000755	0.015445
BRK-B	180	selu	0.1	0.010	0.54	0.004197	0.045921
AVGO	110	elu	0	0.001	0.52	0.000924	0.019610
AMZN	40	selu	0.1	0.001	0.55	0.000431	0.014262
AMGN	150	relu	0.3	0.001	0.53	0.000264	0.010888
ABT	100	elu	0.2	0.001	0.52	0.000256	0.010888
ABBV	240	selu	0.1	0.001	0.52	0.000512	0.015124
AAPL	40	relu	0.2	0.001	0.54	0.000334	0.012071

Table A1 Single stock predictions: Results for GRU, the performance is measured as the Hit-Ratio (HR), Mean Squared Error (MSE) and Mean Absolute Error (MAE). The results are computed for the independent test set.

Stock	Units	Activation	Dropout	Learning-rate	Hit Ratio Test Set	MSE Test Set	MAE Test Set
XOM	240	relu	0.0	0.010	0.51	0.000300	0.011017
WMT	190	relu	0.0	0.010	0.53	0.000206	0.008890
WFC	240	relu	0.0	0.001	0.50	0.000414	0.012681
VZ	70	relu	0.3	0.010	0.52	0.000150	0.008541
V	40	relu	0	0.001	0.52	0.000707	0.015795
UNH	240	relu	0.2	0.010	0.53	0.000333	0.011160
T	70	relu	0.2	0.010	0.53	0.000219	0.009561
PM	100	relu	0.2	0.01	0.53	0.000584	0.014553
PG	210	relu	0.0	0.010	0.53	0.000163	0.008036
PFE	40	relu	0.3	0.010	0.51	0.000198	0.009458
PEP	200	relu	0.0	0.010	0.51	0.000177	0.007889
ORCL	210	relu	0.3	0.010	0.52	0.000286	0.009801
NVDA	150	relu	0.0	0.010	0.49	0.001035	0.022533
MSFT	90	relu	0.1	0.010	0.54	0.000314	0.011096
MRK	70	relu	0.2	0.010	0.51	0.000207	0.009779
MO	110	relu	0.3	0.001	0.51	0.000241	0.010464
MMM	110	relu	0.2	0.010	0.54	0.000234	0.009747
MDT	40	relu	0.1	0.010	0.53	0.000236	0.009822
MCD	90	relu	0.0	0.010	0.54	0.000225	0.008720
MA	40	relu	0.3	0.01	0.57	0.000585	0.015033
KO	40	relu	0.3	0.010	0.52	0.000153	0.007804
JPM	240	relu	0.0	0.010	0.49	0.000359	0.011700
JNJ	110	relu	0.4	0.010	0.52	0.000160	0.007981
INTC	70	relu	0.2	0.010	0.52	0.000434	0.013054
IBM	150	relu	0.4	0.001	0.53	0.000263	0.010319
HON	40	relu	0.3	0.010	0.55	0.000243	0.009597
HD	240	relu	0.4	0.010	0.53	0.000266	0.009962
GSPC	40	relu	0	0.01	0.48	0.000153	0.007180
GOOG	70	relu	0.1	0.01	0.56	0.007180	0.013642
GILD	120	relu	0.0	0.001	0.52	0.000315	0.012344
GE	60	relu	0.3	0.001	0.45	0.000599	0.015836
FB	140	relu	0	0.001	0.53	0.000684	0.017711
DIS	40	relu	0.3	0.010	0.51	0.000276	0.010381
CVX	70	relu	0.2	0.001	0.49	0.000432	0.012261
CSCO	60	relu	0.0	0.010	0.55	0.000322	0.011475
CMCSA	100	relu	0.0	0.010	0.51	0.000259	0.011082
C	150	relu	0.0	0.010	0.50	0.000544	0.014363
BMY	100	relu	0.0	0.010	0.51	0.000294	0.011488
BAC	90	relu	0.0	0.010	0.50	0.000476	0.014148
BA	160	relu	0.0	0.001	0.52	0.000771	0.015515
BRK-B	230	relu	0.4	0.001	0.50	0.000264	0.009920
AVGO	190	relu	0	0.01	0.51	0.000932	0.019737
AMZN	110	relu	0.1	0.010	0.56	0.000432	0.014276
AMGN	130	relu	0.1	0.001	0.50	0.000267	0.010983
ABT	240	relu	0.4	0.010	0.52	0.000256	0.010788
ABBV	170	relu	0.1	0.01	0.48	0.000678	0.016892
AAPL	150	relu	0.0	0.010	0.53	0.000333	0.012061

Table A2 Single stock predictions: Results for RNN, the performance is measured as the Hit-Ratio (HR), Mean Squared Error (MSE) and Mean Absolute Error (MAE). The results are computed for the independent test set.

Stock	Units	Activation	Dropout	Learning-rate	Hit Ratio Test Set	MSE Test Set	MAE Test Set
XOM	70	relu	0.3	0.010	0.51	0.000296	0.011045
WMT	140	relu	0.4	0.001	0.52	0.000213	0.009557
WFC	90	relu	0.4	0.010	0.49	0.000419	0.012711
VZ	160	relu	0.2	0.010	0.52	0.000151	0.008554
V	120	relu	0.1	0.01	0.50	0.001248	0.023014
UNH	80	relu	0.1	0.010	0.53	0.000330	0.011154
T	70	relu	0.3	0.010	0.53	0.000218	0.009604
PM	70	relu	0.3	0.01	0.50	0.001374	0.022662
PG	140	relu	0.3	0.010	0.53	0.000165	0.008059
PFE	130	relu	0.2	0.010	0.51	0.000198	0.009419
PEP	60	relu	0.3	0.010	0.54	0.000172	0.007908
ORCL	40	relu	0.3	0.010	0.55	0.000286	0.009797
NVDA	250	relu	0.0	0.010	0.50	0.001043	0.022721
MSFT	150	elu	0.4	0.010	0.51	0.000316	0.011474
MRK	90	relu	0.4	0.010	0.51	0.000207	0.009760
MO	110	relu	0.1	0.010	0.53	0.000239	0.010365
MMM	100	relu	0.2	0.010	0.53	0.000237	0.009763
MDT	50	relu	0.3	0.010	0.53	0.000234	0.009766
MCD	240	relu	0.1	0.010	0.54	0.000221	0.008652
MA	200	relu	0	0.01	0.50	0.000823	0.018945
KO	90	relu	0.3	0.010	0.52	0.000153	0.007809
JPM	100	relu	0.3	0.010	0.51	0.000353	0.011706
JNJ	210	relu	0.4	0.010	0.52	0.000160	0.007932
INTC	60	relu	0.2	0.010	0.53	0.000429	0.013010
IBM	120	relu	0.1	0.010	0.52	0.000262	0.010433
HON	70	relu	0.3	0.001	0.53	0.000254	0.009940
HD	150	relu	0.3	0.010	0.53	0.000263	0.009943
GSPC	90	relu	0.2	0.01	0.55	0.000150	0.007147
GOOG	230	elu	0	0.001	0.50	0.001691	0.030222
GILD	210	relu	0.4	0.010	0.51	0.000320	0.012365
GE	40	relu	0.3	0.010	0.46	0.000595	0.015730
FB	200	relu	0	0.01	0.50	0.001203	0.024348
DIS	60	relu	0.1	0.010	0.51	0.000277	0.010433
CVX	240	relu	0.1	0.010	0.50	0.000431	0.012344
CSCO	170	relu	0.4	0.010	0.54	0.000322	0.011496
CMCSA	200	relu	0.1	0.010	0.52	0.000256	0.011024
C	210	relu	0.1	0.010	0.50	0.000540	0.014344
BMY	100	relu	0.2	0.010	0.53	0.000290	0.011405
BAC	160	relu	0.2	0.010	0.51	0.000470	0.014136
BA	110	relu	0.3	0.010	0.53	0.000765	0.015678
BRK-B	120	relu	0.4	0.001	0.50	0.000341	0.013548
AVGO	200	relu	0.1	0.01	0.50	0.002243	0.030559
AMZN	100	relu	0.4	0.010	0.56	0.000433	0.014271
AMGN	50	relu	0.3	0.010	0.53	0.000263	0.010872
ABT	100	relu	0.2	0.010	0.53	0.000256	0.010775
ABBV	140	relu	0	0.01	0.51	0.000904	0.021489
AAPL	230	selu	0.3	0.010	0.53	0.000336	0.012108

Table A3 Single stock predictions: Results for DNN, the performance is measured as the Hit-Ratio (HR), Mean Squared Error (MSE) and Mean Absolute Error (MAE). The results are computed for the independent test set.

Stock	Hit Ratio (Baseline)	MSE (Baseline)	MAE (Baseline)	Hit Ratio (SVM)	MSE (SVM)	MAE (SVM)
XOM	0.50	0.000632	0.015859	0.51	0.000345	0.011814
WMT	0.50	0.000457	0.013281	0.53	0.000277	0.011264
WFC	0.50	0.000928	0.018347	0.50	0.000514	0.014825
VZ	0.49	0.000331	0.012486	0.52	0.000195	0.010095
V	0.51	0.001311	0.021221	0.49	0.000556	0.014887
UNH	0.50	0.000756	0.016417	0.53	0.000344	0.012104
T	0.49	0.000477	0.014107	0.50	0.000250	0.010583
PM	0.50	0.001255	0.021944	0.53	0.000642	0.017719
PG	0.50	0.000368	0.011946	0.53	0.000409	0.017270
PFE	0.48	0.000419	0.013470	0.51	0.000336	0.014676
PEP	0.50	0.000429	0.011908	0.49	0.000192	0.008631
ORCL	0.49	0.000709	0.014924	0.54	0.000359	0.008631
NVDA	0.50	0.002350	0.033838	0.53	0.001063	0.023067
MSFT	0.50	0.000781	0.017126	0.51	0.000318	0.011715
MRK	0.49	0.000463	0.014412	0.48	0.000265	0.012012
MO	0.49	0.000480	0.014822	0.53	0.000349	0.013926
MMM	0.50	0.000501	0.014242	0.47	0.000317	0.012964
MDT	0.50	0.000516	0.014465	0.53	0.000255	0.010989
MCD	0.50	0.000507	0.013081	0.55	0.001463	0.035814
MA	0.51	0.001436	0.023700	0.56	0.000832	0.018945
KO	0.49	0.000315	0.011264	0.52	0.000725	0.024353
JPM	0.49	0.000833	0.017217	0.51	0.000418	0.014060
JNJ	0.50	0.000352	0.011799	0.52	0.000847	0.026807
INTC	0.50	0.001056	0.019622	0.47	0.000598	0.017028
IBM	0.50	0.000566	0.015278	0.52	0.000334	0.012556
HON	0.50	0.000523	0.013947	0.46	0.000290	0.011331
HD	0.50	0.000603	0.014446	0.52	0.000330	0.011157
GSPC	0.50	0.000367	0.010748	0.45	0.000938	0.028746
GOOG	0.50	0.001078	0.021360	0.54	0.001465	0.031062
GILD	0.50	0.000689	0.018163	0.51	0.000426	0.015107
GE	0.49	0.001236	0.022112	0.46	0.000769	0.018264
FB	0.51	0.001751	0.026899	0.57	0.001480	0.031083
DIS	0.50	0.000627	0.015541	0.51	0.000350	0.012839
CVX	0.50	0.000973	0.018105	0.50	0.000495	0.013165
CSCO	0.49	0.000777	0.017491	0.51	0.000344	0.012374
CMCSA	0.49	0.000608	0.016686	0.50	0.000296	0.011934
C	0.50	0.001188	0.020651	0.50	0.000579	0.015135
BMY	0.50	0.000600	0.016844	0.50	0.000356	0.013196
BAC	0.48	0.001037	0.020374	0.49	0.000523	0.015533
BA	0.50	0.001372	0.021627	0.48	0.000890	0.018798
BRK-B	0.50	0.000616	0.014766	0.53	0.001133	0.030381
AVGO	0.50	0.002383	0.030533	0.54	0.001359	0.025663
AMZN	0.51	0.000927	0.020713	0.56	0.000548	0.016973
AMGN	0.50	0.000584	0.016245	0.52	0.000346	0.012755
ABT	0.49	0.000574	0.015910	0.45	0.000711	0.021989
ABBV	0.50	0.001128	0.022075	0.46	0.000569	0.015950
AAPL	0.50	0.000742	0.017847	0.54	0.000377	0.013249

Table A4 Single stock predictions: In this table, we can see the results for both the baseline model and Support Vector Machine (SVM). The performance is measured as the Hit-Ratio (HR), the performance is measured as the Hit-Ratio (HR), Mean Squared Error (MSE), and Mean Absolute Error (MAE). The results are computed for the independent test set.

Stock	Start train	End train	Start validation	End validation	Start test	End test
AAPL	1980-12-15	2010-08-26	2010-08-27	2015-08-13	2015-08-14	2020-07-31
ABBV	2013-01-03	2018-09-07	2018-09-10	2019-08-20	2019-08-21	2020-07-31
ABT	1980-03-18	2010-06-21	2010-06-22	2015-07-13	2015-07-14	2020-07-31
AMGN	1983-06-20	2011-04-13	2011-04-14	2015-12-07	2015-12-08	2020-07-31
AMZN	1997-05-16	2014-10-10	2014-10-13	2017-09-05	2017-09-06	2020-07-31
AVGO	2009-08-07	2017-10-30	2017-10-31	2019-03-19	2019-03-20	2020-07-31
BRK-B	1996-05-10	2014-07-11	2014-07-14	2017-07-20	2017-07-21	2020-07-31
BA	1980-01-03	2010-06-02	2010-06-03	2015-07-01	2015-07-02	2020-07-31
BAC	1980-01-03	2010-06-02	2010-06-03	2015-07-01	2015-07-02	2020-07-31
BMY	1980-01-03	2010-06-02	2010-06-03	2015-07-01	2015-07-02	2020-07-31
C	1980-01-03	2010-06-02	2010-06-03	2015-07-01	2015-07-02	2020-07-31
CMCSA	1980-03-18	2010-06-21	2010-06-22	2015-07-13	2015-07-14	2020-07-31
CSCO	1990-02-20	2012-12-17	2012-12-18	2016-10-07	2016-10-10	2020-07-31
CVX	1980-01-03	2010-06-02	2010-06-03	2015-07-01	2015-07-02	2020-07-31
DIS	1980-01-03	2010-06-02	2010-06-03	2015-07-01	2015-07-02	2020-07-31
FB	2012-05-21	2018-07-13	2018-07-16	2019-07-24	2019-07-25	2020-07-31
GE	1980-01-03	2010-06-02	2010-06-03	2015-07-01	2015-07-02	2020-07-31
GILD	1992-01-23	2013-06-12	2013-06-13	2017-01-04	2017-01-05	2020-07-31
GOOG	2004-08-20	2016-08-04	2016-08-05	2018-08-02	2018-08-03	2020-07-31
GSPC	1980-01-03	2010-06-02	2010-06-03	2015-07-01	2015-07-02	2020-07-31
HD	1981-09-23	2010-11-04	2010-11-05	2015-09-18	2015-09-21	2020-07-31
HON	1980-01-03	2010-06-02	2010-06-03	2015-07-01	2015-07-02	2020-07-31
IBM	1980-01-03	2010-06-02	2010-06-03	2015-07-01	2015-07-02	2020-07-31
INTC	1980-03-18	2010-06-21	2010-06-22	2015-07-13	2015-07-14	2020-07-31
JNJ	1980-01-03	2010-06-02	2010-06-03	2015-07-01	2015-07-02	2020-07-31
JPM	1980-03-18	2010-06-21	2010-06-22	2015-07-13	2015-07-14	2020-07-31
KO	1980-01-03	2010-06-02	2010-06-03	2015-07-01	2015-07-02	2020-07-31
MA	2006-05-26	2017-01-12	2017-01-13	2018-10-19	2018-10-22	2020-07-31
MCD	1980-01-03	2010-06-02	2010-06-03	2015-07-01	2015-07-02	2020-07-31
MDT	1980-01-03	2010-06-02	2010-06-03	2015-07-01	2015-07-02	2020-07-31
MMM	1980-01-03	2010-06-02	2010-06-03	2015-07-01	2015-07-02	2020-07-31
MO	1980-01-03	2010-06-02	2010-06-03	2015-07-01	2015-07-02	2020-07-31
MRK	1980-01-03	2010-06-02	2010-06-03	2015-07-01	2015-07-02	2020-07-31
MSFT	1986-03-14	2011-12-19	2011-12-20	2016-04-12	2016-04-13	2020-07-31
NVDA	1999-01-25	2015-03-17	2015-03-18	2017-11-20	2017-11-21	2020-07-31
ORCL	1986-03-13	2011-12-19	2011-12-20	2016-04-12	2016-04-13	2020-07-31
PEP	1980-01-03	2010-06-02	2010-06-03	2015-07-01	2015-07-02	2020-07-31
PFE	1980-01-03	2010-06-02	2010-06-03	2015-07-01	2015-07-02	2020-07-31
PG	1980-01-03	2010-06-02	2010-06-03	2015-07-01	2015-07-02	2020-07-31
PM	2008-03-18	2017-06-27	2017-06-28	2019-01-14	2019-01-15	2020-07-31
T	1983-11-22	2011-05-23	2011-05-24	2015-12-24	2015-12-28	2020-07-31
UNH	1984-10-18	2011-08-15	2011-08-16	2016-02-08	2016-02-09	2020-07-31
V	2008-03-20	2017-06-27	2017-06-28	2019-01-14	2019-01-15	2020-07-31
VZ	1983-11-22	2011-05-23	2011-05-24	2015-12-24	2015-12-28	2020-07-31
WFC	1980-01-03	2010-06-02	2010-06-03	2015-07-01	2015-07-02	2020-07-31
WMT	1980-01-03	2010-06-02	2010-06-03	2015-07-01	2015-07-02	2020-07-31
XOM	1980-01-03	2010-06-02	2010-06-03	2015-07-01	2015-07-02	2020-07-31

Table A5 Train, Validation and Test periods for the univariate study.

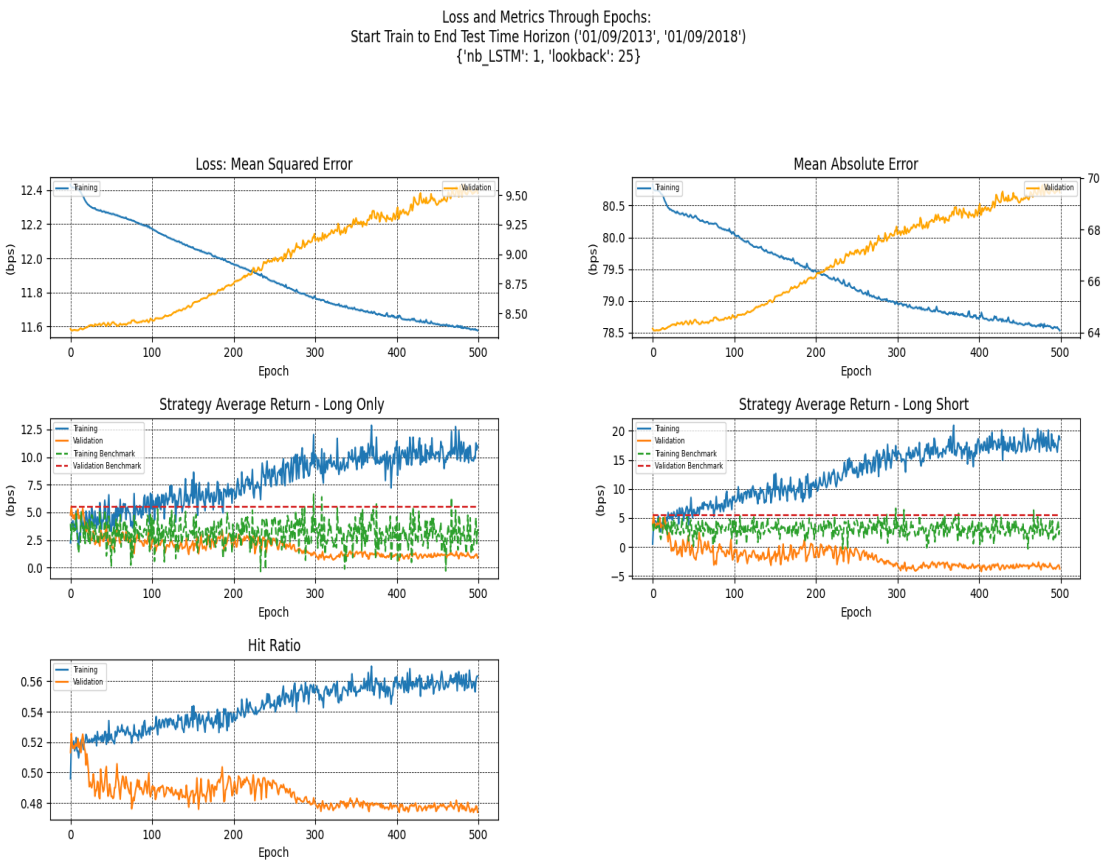

Figure 11. Training and validation metrics for a single LSTM cell as hidden layer with lookback 25 on the start training to end testing horizon 2013-09-01 to 2018-09-01

B. Multiple Stocks Experiment

Network Type	Long Short Term Memory
Hidden Layers	1 Unit LSTM Layer , Relu Activation, Random Uniform Initialization
Output Layer	Dense, tanh Activation, Random Uniform Initialization
Loss Function	Mean Squared Error
Optimizer	Adam, Batch Size 128

Table A6 Architecture Details Collective Stock Prediction

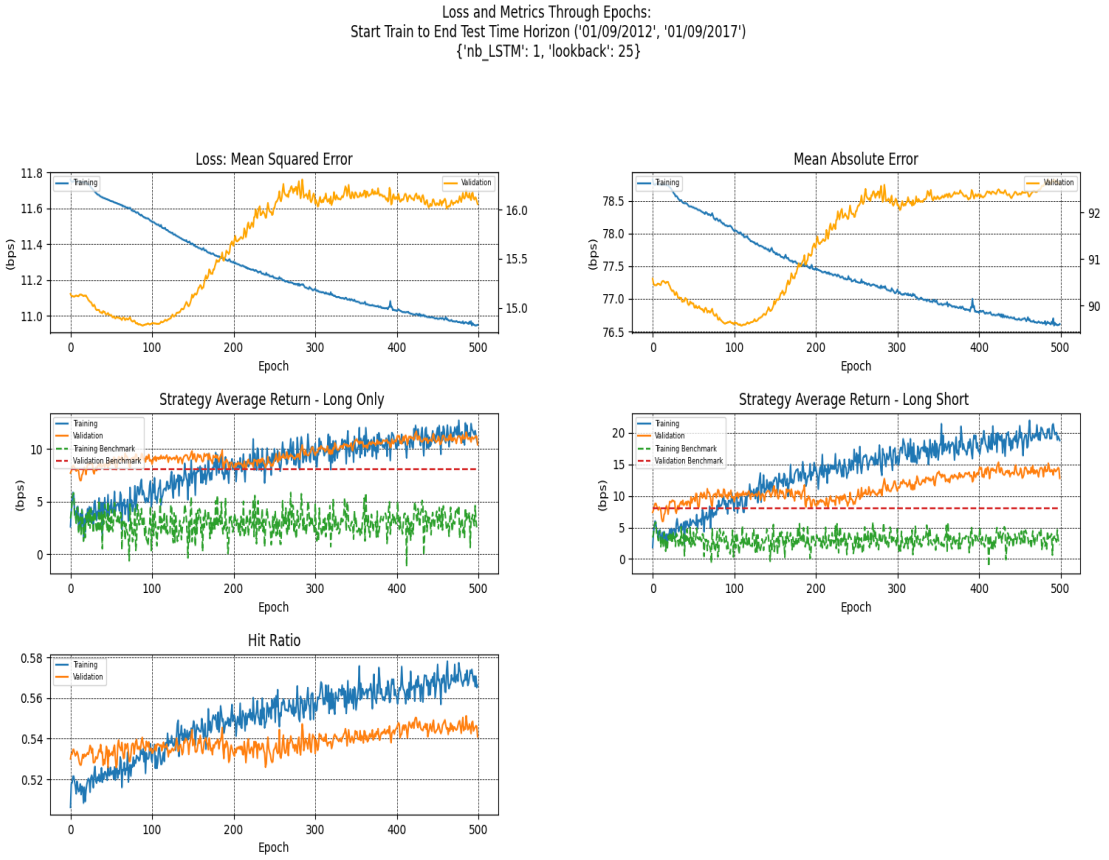

Figure 12. Training and validation metrics for a single LSTM cell as hidden layer with lookback 25 on the start training to end testing horizon 2012-09-01 to 2017-09-01

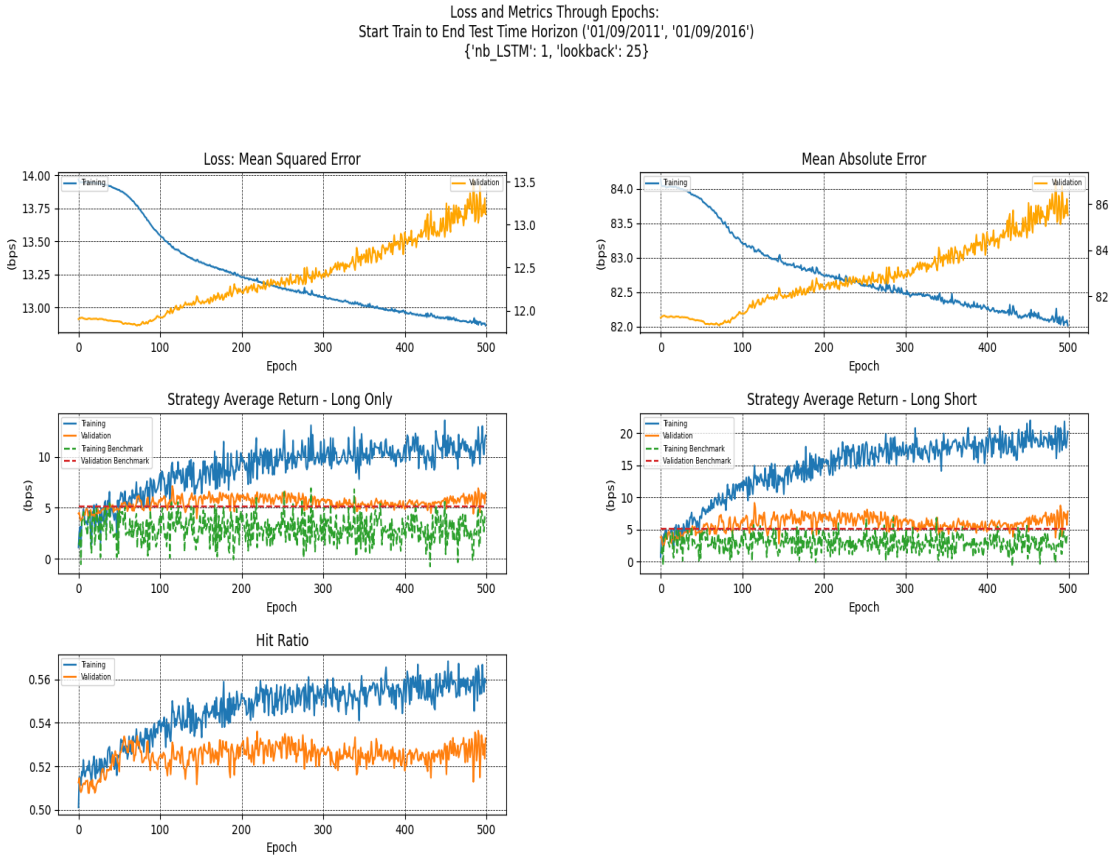

Figure 13. Training and validation metrics for a single LSTM cell as hidden layer with lookback 25 on the start training to end testing horizon 2011-09-01 to 2016-09-01

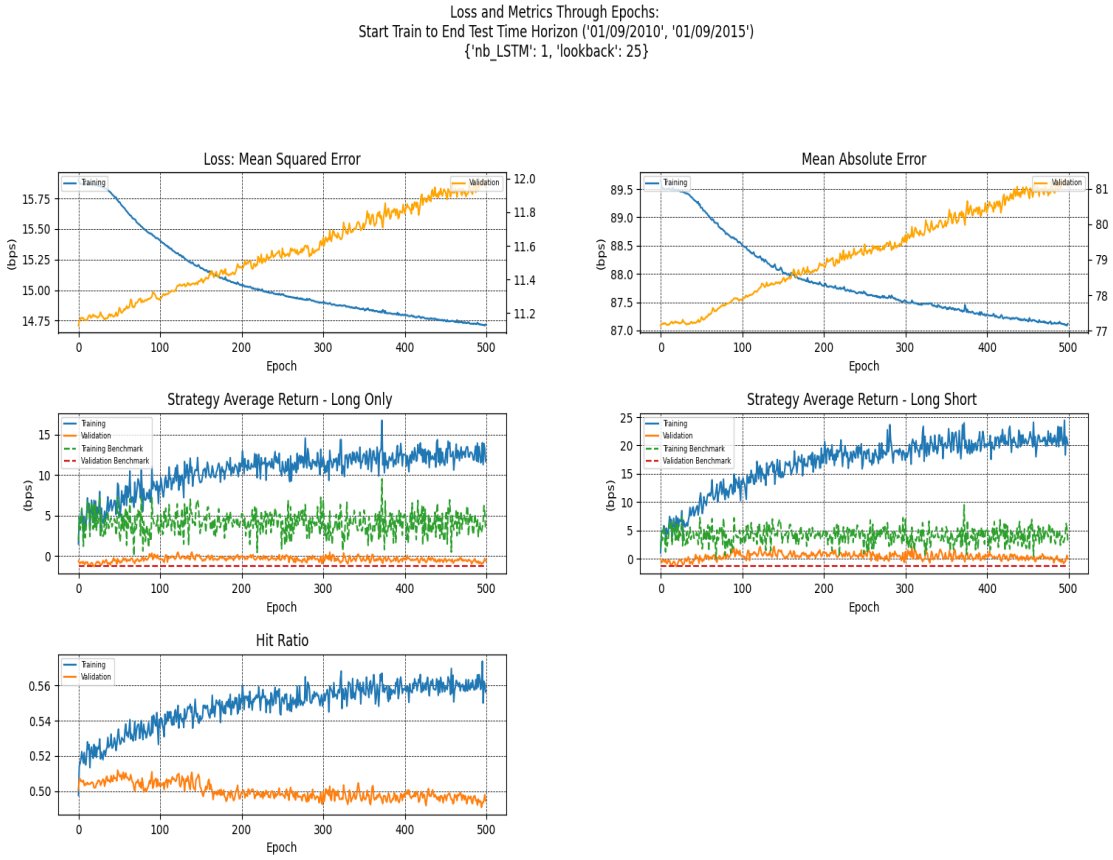

Figure 14. Training and validation metrics for a single LSTM cell as hidden layer with lookback 25 on the start training to end testing horizon 2010-09-01 to 2015-09-01

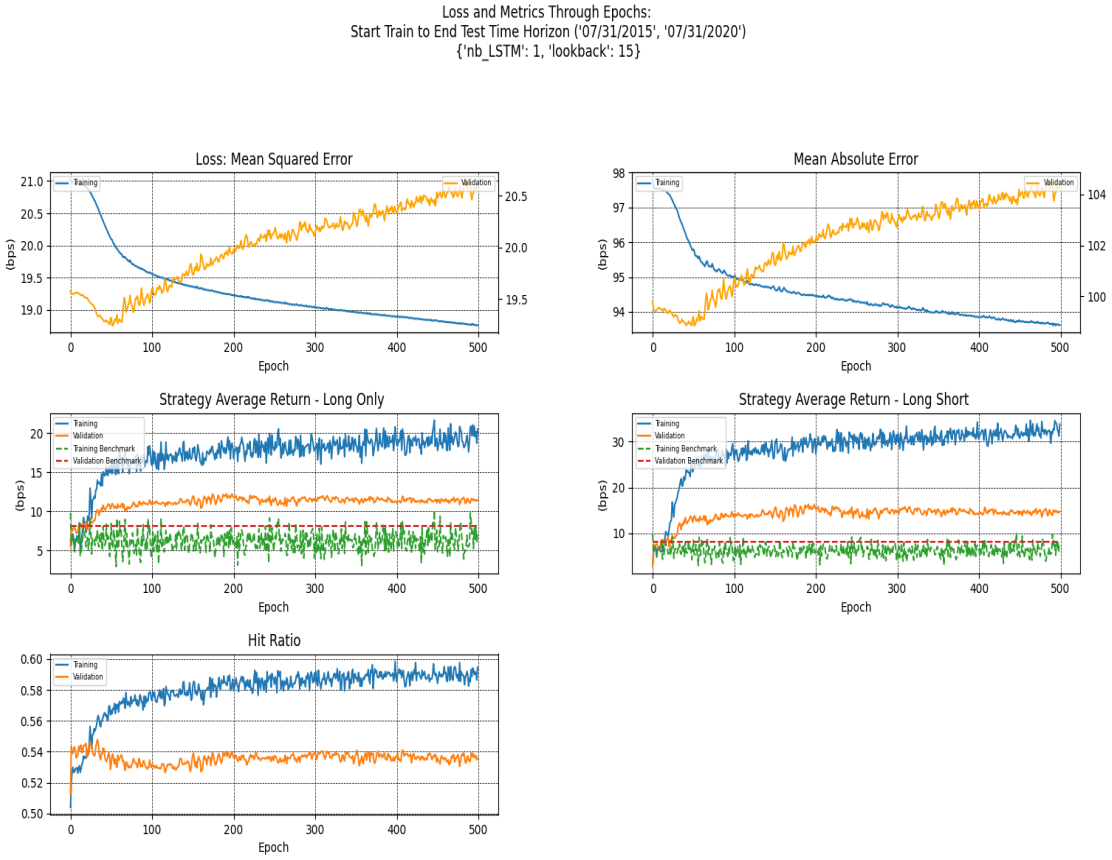

Figure 15. Training and validation metrics for a single LSTM cell as hidden layer with lookback 15 on the start training to end testing horizon 2015-07-31 to 2020-07-31

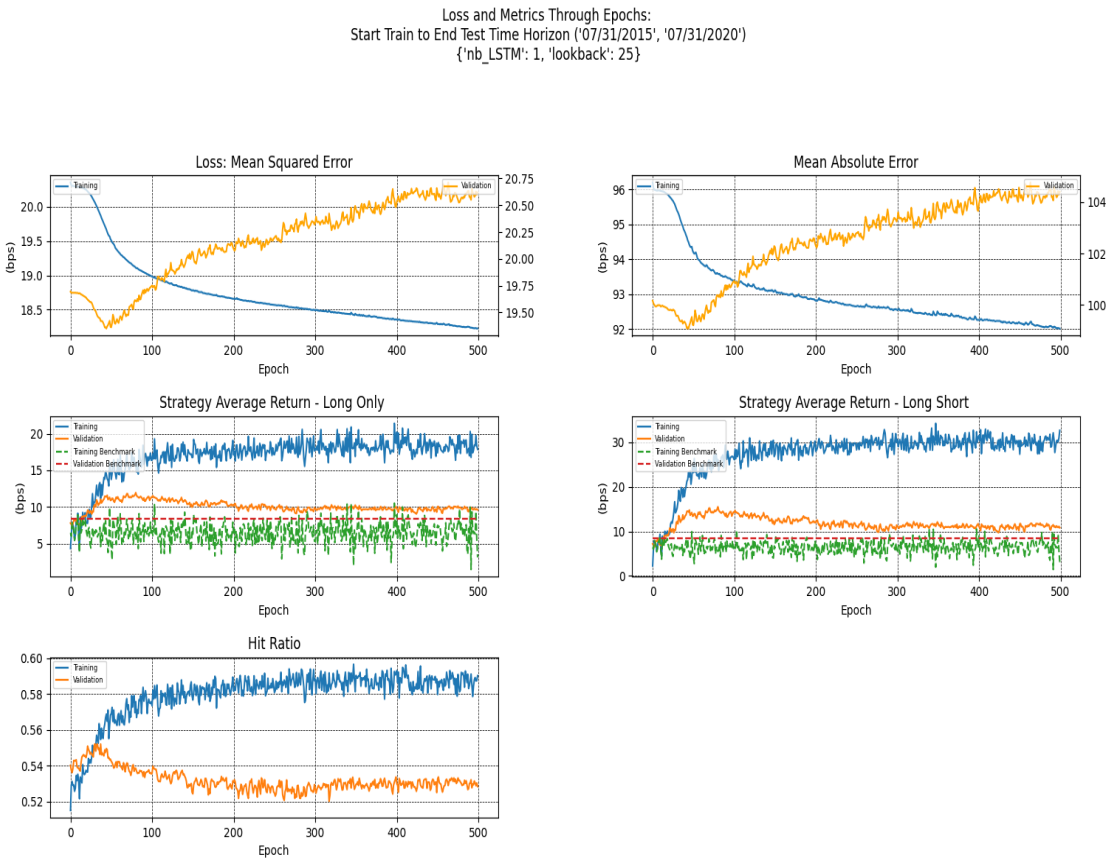

Figure 16. Training and validation metrics for a single LSTM cell as hidden layer with lookback 25 on the start training to end testing horizon 2015-07-31 to 2020-07-31

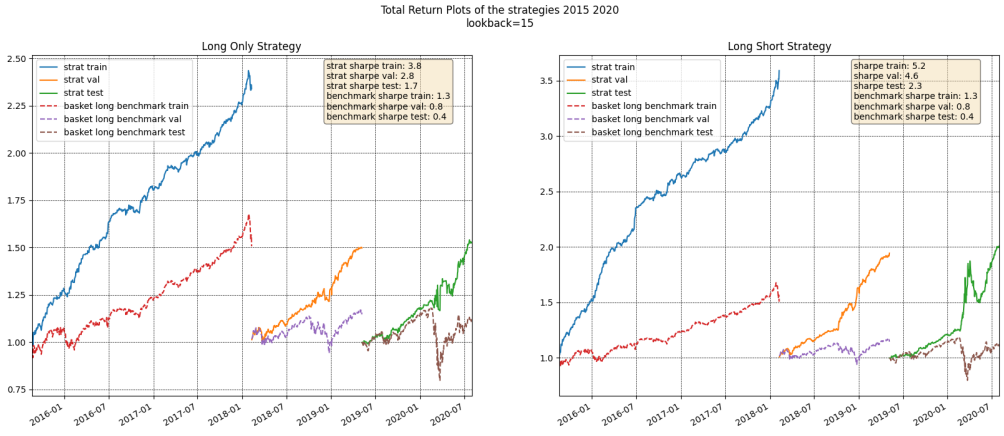

Figure 17. Cumulative return of the strategies compared to the benchmark consisting in holding the basket long for the full period for a single LSTM cell as hidden layer with lookback 15 on the start training to end testing horizon 2015-07-31 to 2020-07-31

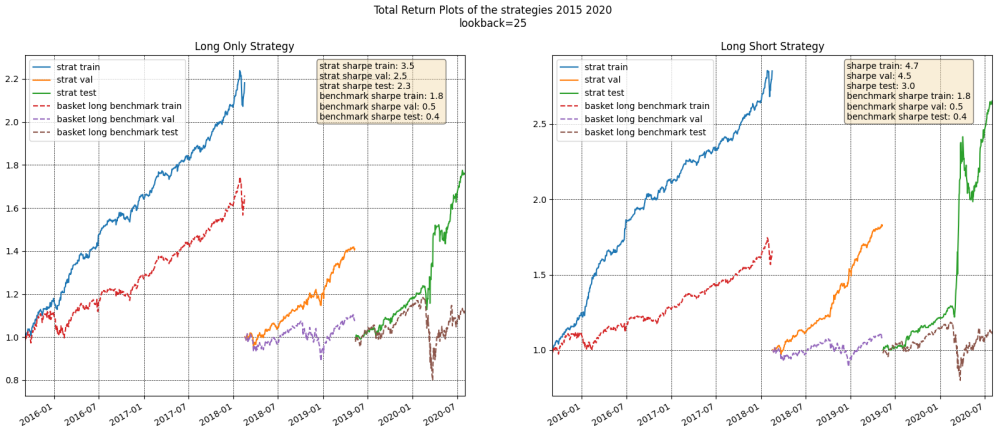

Figure 18. Cumulative return of the strategies compared to the benchmark consisting in holding the basket long for the full period for a single LSTM cell as hidden layer with lookback 25 on the start training to end testing horizon 2015-07-31 to 2020-07-31

C. Implementation and Hardware

For the computations, we used the Python programming language and the Keras Deep Learning library. Keras uses tensors with strong GPU acceleration. The GPU computations were performed on an NVIDIA GeForce GTX 1080 Ti and an NVIDIA GeForce GTX 1070 GDDR5 card on two separate machines.

REFERENCES

[1] de Prado, M. Machine Learning for Asset Managers. ISBN-13 : 978-1108792899 Elements in Quantitative Finance. Cambridge University Press, 2020.

[2] Dixon, M., Halperin, I., and Bilokon, P. *Machine Learning in Finance: From Theory to Practice*. ISBN 978-3030410674 Springer science, business media New York, inc , 2020.

[3] Coqueret, G. and Guida,T. Machine Learning for Factor Investing: R Version. ISBN-13 : 978-0367545864 Chapman and Hall, 2020.

[4] Ian Goodfellow and Yoshua Bengio and Aaron Courville. Goodfellow et al. *Deep Learning*. MIT Press, `http://www.deeplearningbook.org`, 2016.

[5] Alex Graves. *Supervised sequence labelling with recurrent neural networks*. ebook: ISBN 978-3-642-24797-2; Springer-Verlag Berlin Heidelberg, 2012.

[6] Ilya Sutskever. *Training recurrent neural networks: Data mining, inference and prediction*. PhD thesis, department of computer science, University of Toronto, 2013.

[7] Sepp Hochreiter; Jurgen Schmidhuber. *Long short-term memory*. Neural computation 9, 1735-1780, (1997).© 1997 Massachusetts Institute of Technology.

[8] Cho, Kyunghyun; van Merrienboer, Bart; Gulcehre, Caglar; Bahdanau, Dzmitry; Bougares, Fethi; Schwenk, Holger; Bengio, Yoshua (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". arXiv:1406.1078.

[9] *Weiss, Gail; Goldberg, Yoav; Yahav, Eran (2018). "On the Practical Computational Power of Finite Precision RNNs for Language Recognition". arXiv:1805.04908 [cs.NE].*

[10] Ashish Vaswani and Noam Shazeer and Niki Parmar and Jakob Uszkoreit and Llion Jones and Aidan N. Gomez and Lukasz Kaiser and Illia Polosukhin Attention Is All You Need, Arxiv:1706.03762. 2017

[11] Alonso, Miquel N. and Batres-Estrada, Gilberto and Moulin, Aymeric Deep Learning in Finance: Prediction of Stock Returns with Long Short-Term Memory Networks booktitle = Big Data and Machine Learning in Quantitative Investment, pages = 251-277, doi = 10.1002/9781119522225.ch13 Wiley,2018

[12] Matthew F Dixon *Industrial Forecasting with Exponentially Smoothed Recurrent Neural Networks* ArXiv:2004.04717 April 2020

[13] Lim, Bryan and Zohren, Stefan and Roberts, Stephen, Enhancing Time Series Momentum Strategies Using Deep Neural Networks (April 9, 2019). The Journal of Financial Data Science, Fall 2019, https://jfds.pm-research.com/content/1/4/19, Available at SSRN: https://ssrn.com/abstract=3369195 or http://dx.doi.org/10.2139/ssrn.3369195

[14] Thomas Fischer; Christopher Krauss. *Deep Learning with long short-term memory networks for financial market predictions.* ISSN 1867-6767, Friedrich-Alexander-Universität Erlangen-Nürnberg, Institute for economics. `https://www.iwf.rw.fau.de/research/iwf-discussion-paper-series/`.

[15] Weinan E and Chao Ma and Stephan Wojtowytsch and Lei Wu Towards a Mathematical Understanding of Neural Network-Based Machine Learning: what we know and what we don't, eprint=2009.10713, arXiv 2020

[16] Sang Il Lee, Seong Joon Yoo. 2017. A deep efficient frontier method for optimal investments. *Expert systems with applications*, September.

[17] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, Noam Shazeer. *Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks.* Google Research, Mountain View, CA, USA bengio,vinyals,ndjaitly,noam@google.com.

[18] F. M. Bianchi, M. Kampffmeyer, E. Maiorino, R. Jenssen. *Temporal Overdrive Recurrent Neural Network, arXiv preprint arXiv:1701.05159.*

[19] François Chollet et al. *Keras.* GitHub, 2015. `https://github.com/keras-team/keras`.

[20] J. L. Elman. *Language as a dynamical system, Mind as motion: Explorations in the dynamics of cognition (1995) 195{223}.*
@misc{chollet2015keras, title={Keras}, author={Chollet, François and others}, year={2015}, publisher={GitHub}, howpublished={`https://github.com/keras-team/keras`},

[21] Lisha Li; Kevin Jamieson; Giulia DeSalvo; Aifshin Rostamizadeh; Ameet Talwalkar. *Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization.*

[22] Razvan Pascanu; Tomas Mikolov; Yoshua Bengio. *On the difficulty of training recurrent neural networks.* Proceedings of the 30^{th} international conference on machine learning, Atlanta, Georgia, USA, 2013. JMLR W&CP volume 28. Copyright by the author(s) 2013.

[23] Quandl. `https://www.quandl.com/`

[24] F. Rosenblatt The perceptron: a probabilistic model for information storage and organization in the brain Psychological review, Vol 65, No. 6, 1958.

[25] Nits Srivastava; Geoffrey Hinton; Alex Krizhevsky; Ilya Sutskever; Ruslan Salakhutdinov. *Dropout: A simple way to prevent neural networks from overfitting.* Journal of machine learning research 15 (2014) 1929-1958, 2014.