2022 IEEE Symposium on Computational Intelligence for Financial Engineering and Economics (CIFEr) | 978-1-6654-4234-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/CIFEr52523.2022.9776177

Customized Stock Return Prediction with Deep Learning

Christopher Felder
Department of Banking
University of Tübingen
Tübingen, Germany
christopher.felder@uni-tuebingen.de

Stefan Mayer
Department of Marketing
University of Tübingen
Tübingen, Germany
stefan.mayer@uni-tuebingen.de

***Abstract*—In finance, researchers so far use standard loss functions such as mean squared error when training artificial neural networks for return prediction. However, from an investor's perspective, prediction errors are ambiguous: in practice, the investor prefers to see underprediction of portfolio returns rather than overprediction, as the former implies higher realized returns and thus financial benefits. We present a loss function customized to this behavior and test it based on Long Short-Term Memory (LSTM) models, the state-of-the-art tools in time series analysis. Our model learns unique signals, predicts returns more cautiously, and improves profit chances over the standard LSTM and reversal signals. Daily and weekly revised portfolios achieve on average five percentage points higher annualized returns. We show that our loss function is robust to market sentiment and beneficial in nonlinear optimization.**

***Index Terms*—Time series analysis, LSTM, loss function, return prediction**

I. Introduction

An extensive literature examines trend-based return predictability and identifies numerous predictors widely used in practice [1], [2]. In recent years, dynamic self-learning approaches have revolutionized technical analysis. One powerful deep learning tool well suited for time series problems is the Long Short-Term Memory (LSTM) proven successful in many return prediction problems, see [3]–[7] – to name a few.

We reconsider return prediction using LSTM from an economic perspective and link the optimization problem to characteristics of portfolio management in practice. Investors have certain interests when trading shares, and one of them is achieving return targets. An investor predicting a portfolio return of 5% would be disappointed if the actual return was 3%, but would be happy if the actual return was 7%. Standard optimization techniques handle those two prediction errors simultaneously, and both the positive and negative errors receive the same penalty, even though the investor actually suffers from only one error.

A loss function tailored to the economic properties of return prediction can improve the economic benefit from predictions. For instance, the value-weighted loss function in [8] forces networks to adjust parameters focusing on the prediction performance for the main drivers of stock markets. The loss function proposed in our work distinguishes between 'good' and 'bad' prediction losses. 'Good' losses occur when the predicted portfolio return is lower than the realized return, 'bad' losses describe the opposite case. The economic objective is to minimize both 'good' and 'bad' prediction losses, with 'good' losses preferred over 'bad' losses. Our framework is based on a weighted mean squared error, where the weight is a function of the loss class and penalizes 'bad' losses up to twice as hard as 'good' losses. Applied to our example, we would penalize the prediction error up to twice as hard when the realized return is 3% than when it is 7%. We discuss a binary weight function that only knows the maximum and minimum penalties, and a sigmoid weight function that distributes weights continuously between the maximum and minimum penalties.

We benchmark our new approach with a standard LSTM as well as short-term reversal, one of the most popular and robust trend-based predictors proven competitive with deep learning [9], whose signals are probably similar to LSTM signals [3], [4], [8]. We arrive at five findings: First, our customized LSTMs underpredict positive returns rather than overpredict them. Investors are less likely to face situations in which the realized portfolio return is lower than the predicted return. Second, we report improved portfolio performance when tested for different prediction horizons. Third, in addition to well-known reversal signals, we extract new unique signals that contain superior predictive information. Fourth, an investor benefits from the loss functions in the nonlinear environment of the LSTM. Fifth, our loss function contributes positively to portfolio returns throughout changing market sentiments.

II. LSTM networks

A. The Long Short-Term Memory architecture

The LSTM is an artificial recurrent neural network architecture introduced by [10]. While traditional feedforward neural networks have no way to extract temporally encoded information from time series data, recurrent networks use time components based on feedback connections with neurons of the same or previous layers. A hidden state transfers information contained in cell outputs across time steps and facilitates stepwise processing of entire sequences to connect information between time steps. However, conventional recurrent networks suffer from short-term memory in practice. The greater the time gap between the occurrence of an information and when it is needed, the less able recurrent networks are to remember and link the information. LSTMs overcome these deficits by using two memory states: The cell state s serves as a long-term memory and carries all information learned in the past when processing the sequence. The hidden state h serves as short-term memory and carries the cell state adapted by information from the recent steps. Instead of updating all parameters at each step of the sequence, multiple layers acting as gates protect the cell state from unnecessary updates. Information that is far in the past survives, but new information still has a chance to remove or update the cell knowledge.

An LSTM cell has three gates: The forget gate f processes the vector of hidden states h_{t-1} and the vector of elements of the input sequence x_t through a logistic layer σ to determine which values of the vector of cell states s_{t-1} to remove. f_t in (1) is a vector of activation values for each element in s_{t-1}, where the vector w_f depicts the weights and b_f the biases. The input gate i controls that only the necessary information from h_{t-1} and x_t pass to s_{t-1}. The vector i_t in (2) activates elements in h_{t-1} and x_t according to their relevance. Equation (3) uses a $tanh$ layer to compute the vector of candidate values $\tilde{s}_t$ that could be added to s_{t-1}. The new cell state s_t in (4) equals the sum of $\tilde{s}_t$ scaled by the activation values i_t and the information survived from s_{t-1}. The output gate o manages the information flow to the hidden state. The vector o_t in (5) determines which information to pass to the new hidden state. The new hidden state h_t in (6) is s_t filtered by the important short-term information carried by h_{t-1} and x_t.

$$f_t = \sigma(w_{f,x} \cdot x_t + w_{f,h} \cdot h_{t-1} + b_f) \tag{1}$$

$$i_t = \sigma(w_{i,x} \cdot x_t + w_{i,h} \cdot h_{t-1} + b_i) \tag{2}$$

$$\tilde{s}_t = tanh(w_{\tilde{s},x} \cdot x_t + w_{\tilde{s},h} \cdot h_{t-1} + b_{\tilde{s}}) \tag{3}$$

$$s_t = f_t \circ s_{t-1} + i_t \circ \tilde{s}_t \tag{4}$$

$$o_t = \sigma(w_{o,x} \cdot x_t + w_{o,h} \cdot h_{t-1} + b_o) \tag{5}$$

$$h_t = o_t \circ tanh(s_t) \tag{6}$$

B. Network configuration

We propose an LSTM network that processes one-day stock returns $r_{j,t}$ and market returns $r_{m,t}$. Our approximation to the market portfolio is an equally weighted portfolio holding the S&P 500 and the Nasdaq Composite. On day t, the one-day return on asset j with close prices p is $r_{j,t} = \frac{p_{j,t}}{p_{j,t-1}} - 1$.

The network architecture comprises an input layer with two features $(r_{j,t}, r_{m,t})$ and 60 days time dimension, one single hidden layer with 32 LSTM cells and a one-node dense output layer $(r_{j,t})$, which is similar to [3], [7]. The time dimension varies by researchers from five to 60 days in [9] up to one year in [7]. We choose a window of 60 days, accounting for the fact that LSTMs retrieve the most important information from recent time steps. At market close on day t, the network processes the sequence $[r_{j,t-59}, r_{m,t-59}, r_{j,t-58}, r_{m,t-58}, \cdots, r_{j,t}, r_{m,t}]$ and predicts the next one-day return $r_{j,t+1}$.

We regularize our model in three ways. First, we prohibit cell states from communicating across training batches. In a stateful LSTM, memory transfers information between sequences of different batches so that states persist for a full training epoch. Our stateless configuration allows states to transfer information between sequences of the same batch, but resets memory after one batch. Second, we allow the network to shuffle samples within batches to ensure generalizability across batches. Batches thus contain randomly picked sequences from different shares at different time points. Third, we apply model averaging using dropout. Dropout randomly ignores a certain fraction of neurons in hidden and visible layers. We use a dropout ratio of 30% and a batch size of 8.

III. Short-term reversal signals

A well-known trend phenomenon is short-term reversal. Stocks with relatively poor (strong) performance in the previous short-term period, e.g., one month, are likely to produce positive (negative) abnormal returns in the following period [11]–[14]. The most common explanations are investor behavioral bias leading to over- or underreaction to events [15], [16], and size [17] and liquidity effects [18]. Research on reversal phenomena assumes that past performances affect investors' behavior and contain predictive signals for future returns, contrasting with the efficient market hypothesis [19].

The literature documents a close relationship between reversal signals and LSTM signals. In [3], LSTMs buy (sell) stocks that previously exhibited sharp declining (increasing) returns. Other researchers employ LSTMs utilizing reversal patterns [7] and point out that LSTMs extract the most important information for predictions from reversal signals [4], [8]. Another well-documented trend signal is momentum [14]. Since LSTMs do not extract momentum signals [3] and momentum portfolio returns are less competitive with LSTMs [9], we choose to limit our financial benchmark to reversal.

We follow [9] and compare our models with the month reversal (MR) and the week reversal (WR), which take into account the returns achieved in the last month and week, respectively. The midweek reversal (MWR) focuses on two-day returns. A reversal strategy buys (sells) the shares with the largest negative (positive) return in the previous period.

IV. Related work

We build our methodology on the work of [3] and [9], both of which have analyzed a US stock sample similar to ours. While [3] focus on return classification and compare LSTMs to memory-free classification methods, [9] use convolutional neural networks and document improvements in prediction quality and portfolio performance over WR and MR. Our motivation stems from [20], who discuss that the return predictability of LSTMs weakens and portfolio returns lower when taking into account investor constraints.

Although the invention of LSTM dates back more than two decades, with the increasing availability of computational power and data, financial research has discovered LSTM-driven approaches to model time series data only a few years ago [21]. The literature discusses return predictions with LSTMs that disentangle technical trading indicators [22],

[23], process high-frequency data [5], [24], improve prediction quality over ARIMA or feedforward neural networks [6], [25], detect buy and sell signals from market data [3], [26], model the stochastic discount factor and explore economic state processes [4], trace reversal patterns in pairs trading [7], and include additional layers with attention mechanisms [27].

Our work contributes to this literature with a novel approach that links the learning with LSTM networks to the actual economic impact of return prediction errors on portfolio performance. Recent literature debates models that regard prediction errors from a market perspective, e. g., weighting prediction errors by market capitalization [8] or length of observation period [4]. We show that considering the loss function from an investor perspective improves the trading experience and provides investors with more reliable return forecasts and a lower number of missed portfolio return targets.

V. A TRADING-CUSTOMIZED LOSS FUNCTION

As part of the optimization algorithm, the loss function used to train a deep neural network plays a pivotal role. The loss of a prediction determines by how much the network adjusts the weights to reduce the loss of the next prediction. A large loss is associated with large adjustments. Our goal is to make the loss function distinguish between positive and negative prediction errors. In other words, the network should adjust weights more when the portfolio actually suffers from a prediction error, i.e., realized returns are lower than predicted.

In practice, where investors can both buy and short shares, there are four cases of prediction losses. We assume an investor who buys shares with a positive return prediction ($\hat{r}_{j,t} > 0$) and short sells shares with a negative return prediction ($\hat{r}_{j,t} < 0$). The prediction error is $\hat{r}_{j,t} - r_{j,t}$. Fig. 1 illustrates the change in returns on portfolio p given a loss type for long and short investments. Prediction errors in quadrants II and IV are associated with increasing portfolio returns: in the case of a long (short) position, investors would be happy to see a negative (positive) prediction error because the portfolio return increases unexpectedly. The investor benefits from this 'good' prediction error. In quadrants I and III, the investor faces unexpectedly lower portfolio returns ('bad' errors).

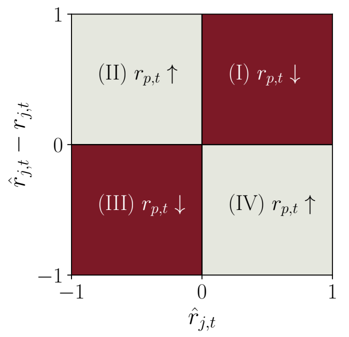

Fig. 1. Change in portfolio returns $r_{p,t}$ given a return prediction error. Red (gray) quadrants indicate decreased (increased) portfolio returns.

We use the term 'unexpected portfolio losses' to describe a situation in which the realized portfolio return is less than the predicted portfolio return, but can still be positive. A predicted return of 5% with a realized return of 3% means an unexpected loss of 2%. 'Unexpected portfolio profits' describe situations where the realized return is greater than the predicted return.

Next, we need a framework that considers prediction losses according to Fig. 1. The most common regression loss function is the mean squared error (MSE) that treats any prediction loss equally. Our proposed loss function is a weighted MSE ($WMSE$) in the style of [8] that weights prediction errors according to their loss type. The $WMSE$ of a prediction set for n shares and z time steps is

$$WMSE = \frac{1}{nz} \sum_{j=1}^{n} \sum_{t=1}^{z} w(\hat{r}_{j,t}, r_{j,t}) \cdot (\hat{r}_{j,t} - r_{j,t})^2 \qquad (7)$$

where the weight $w(\hat{r}_{j,t}, r_{j,t})$ is a function of the loss type determined by $\hat{r}_{j,t}$ and $r_{j,t}$. $w(\hat{r}_{j,t}, r_{j,t})$ must satisfy two requirements: First, the loss function should penalize small 'bad' errors heavier than small 'good' errors. Second, large 'bad' errors should receive the maximum weight and large 'good' errors should receive the minimum weight.

Before configuring the weight function, we have to define a numerical range for the minimum and maximum weights. Since our economic motivation is not to minimize only the 'bad' losses, but both the 'good' and the 'bad' losses, preferring the 'good' over the 'bad' losses, we propose to use the range $w(\hat{r}_{j,t}, r_{j,t}) \in [1, 2]$. A prediction error that leads to an unexpected loss in portfolio value is weighted at most twice as much as an error that leads to an unexpected profit.

There are two ways to meet the loss function requirements: Either using a continuous or a binary weight distribution. A binary weighting function forces the model to distinguish exclusively between 'bad' and 'good' errors. Small negative errors are penalized the same as large negative errors, and the difference in penalty between small positive and small negative errors equals the difference for large positive versus large negative errors. The binary weight function $w^{bin}(\hat{r}_{j,t}, r_{j,t})$ is

$$w^{bin}(\hat{r}_{j,t}, r_{j,t}) = \begin{cases} 1.5 + sign(\hat{r}_{j,t})(0.5) & \text{for} \quad \hat{r}_{j,t} - r_{j,t} > 0 \\ 1.5 - sign(\hat{r}_{j,t})(0.5) & \text{for} \quad \hat{r}_{j,t} - r_{j,t} \leq 0 \end{cases} \qquad (8)$$

and can only take 1 or 2. $sign(\hat{r}_{j,t})$ is the sign of the predicted return $\hat{r}_{j,t}$ that determines the type of investment. If the investor buys the security ($\hat{r}_{j,t} > 0$), the penalty for a positive prediction error is $1.5 + 0.5 = 2$ and for negative prediction errors $1.5 - 0.5 = 1$. If the investor short-sells the security, the penalty for a positive prediction error is $1.5 - 0.5 = 1$.

A continuous function provides a more economical understanding. Small prediction losses have less impact on the investor's portfolio than large losses. The model learns that a large loss is worse than a small loss and penalizes very small positive losses similarly with very small negative losses. The continuous weighting function $w^{log}(\hat{r}_{j,t}, r_{j,t})$ is

$$w^{log}(\hat{r}_{j,t}, r_{j,t}) = \frac{1}{1 + e^{-sign(\hat{r}_{j,t})(\hat{r}_{j,t} - r_{j,t})\cdot 100}} + 1 \quad (9)$$

Since the prediction loss is usually a small percentage, we multiply it by 100 to accelerate the weighting. If the investor is long, a loss of 2% (−2%) receives a weight of 1.88 (1.12), while a small loss of 0.1% (−0.1%) receives 1.52 (1.48).

One might wonder how investors' views on prediction errors would affect a model's prediction quality. Our loss function aims to make return predictions more conservative, i.e., to narrow the distribution of return predictions. In a scenario where return overpredictions exceed underpredictions, using our customized loss function improves the network's approximation to the market over standard LSTMs. Table I and Fig. 2 illustrate that while the pure number of overpredictions and underpredictions is roughly balanced, standard LSTMs overpredict (underpredict) positive (negative) returns on average. Thus, teaching a network to predict returns more conservatively can provide a better approximation of the market.

VI. Daily prediction of US stock returns

A. Data

We exploit the Refinitiv database [28] for daily adjusted closing prices of US stocks primarily listed on NYSE or Nasdaq between 1990 and 2021. The set includes 9,289 shares from NYSE and 6,841 shares from Nasdaq. Since we address our work to private investors, we require a minimum market cap of $50 million to ensure liquidity and moderate transaction costs. We standardize returns by subtracting the cross-sectional mean of returns and dividing by the standard deviation.

We follow [9] and split the data set into a shorter period for training and validation (1990 to 2004), with validation accounting for the last quarter of the period, and a longer period for testing (2005 to 2021). Training a model once on the training data and then test it on the test data makes the model more general and meaningful compared to retraining the parameters multiple times over rolling windows as in [3]. The need to repeatedly retrain a model reduces its general applicability. The validation period serves for tuning the network's hyperparameters. We train our models using *keras* [29] and Google's *TensorFlow* [30]. The training runs for 50 epochs.

B. Prediction quality

We group the test set into $\hat{r}_{j,t} \leq 0$ and $\hat{r}_{j,t} > 0$ to assess the behavior of the network for long and short investments. Table I reports the average prediction errors and the share of negative prediction errors. LSTM^{bin} (LSTM^{log}) denotes the LSTM trained with the binary (logistic) weight function. LSTM^{mse} represents the standard LSTM trained with the MSE.

In our two customized LSTMs, the average prediction errors are greater in the short position than in the long position ($-0.01 > -0.07$ and $0.08 > 0.03$), i.e., the models favor 'good' losses when buying shares over 'bad' losses when selling shares. The proportion of unexpected profits on long positions exceeds the proportion of unexpected losses on short positions. In contrast, the average error of the MSE model is greater in the long position than in the short position, resulting in larger unexpected losses on both types of investments. Investors face relatively more 'bad' losses.

Next, we analyze the average prediction error by $\hat{r}_{j,t}$ percentile. Fig. 2 is an empirical illustration of the theoretical model in Fig. 1. The customized LSTMs largely squeeze losses into quadrants II and IV. The 'bad' prediction errors in quadrants I and III are low compared to the 'good' errors in quadrants II and IV. The negative slope indicates the preference for 'good' errors in both types of investments. The MSE model has 'bad' errors in about three-quarters of $\hat{r}_{j,t}$ percentiles. In particular, strong buy and strong sell classifications pose significant unexpected losses.

Next, we convert our prediction task into a classification problem with buy ($\hat{r}_{j,t} > 0$) and sell ($\hat{r}_{j,t} < 0$) signals. Reversals classify shares with preceding negative (positive) returns as buy (sell). A true classification is a buy (sell) classification followed by a positive (negative) realized return.

TABLE I
Prediction error metrics

Model	$\hat{r}_{j,t} \leq 0$	$\hat{r}_{j,t} > 0$	All
Average prediction error* (%)			
LSTM^{bin}	-0.01	-0.07	-0.04
LSTM^{log}	0.08	0.03	0.05
LSTM^{mse}	-0.02	0.18	0.08
Share of negative prediction errors			
LSTM^{bin}	0.50	0.51	0.50
LSTM^{log}	0.48	0.49	0.49
LSTM^{mse}	0.49	0.47	0.48

*The prediction error is $\hat{r}_{j,t} - r_{j,t}$.

TABLE II
Prediction quality

Model	Accuracy	Correlation
LSTM^{bin}	52.24	4.49
LSTM^{log}	52.29	4.58
LSTM^{mse}	52.17	4.35
MR	51.93	3.87
WR	52.16	4.33
MWR	52.20	4.41

(MR) Month reversal (WR) Week reversal (MWR) Midweek reversal

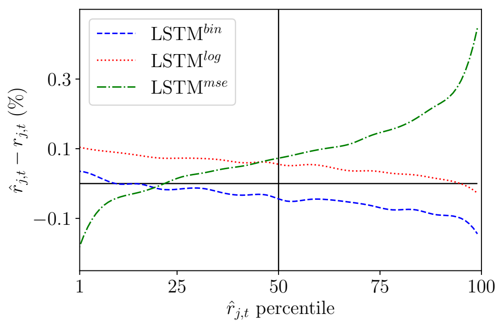

Fig. 2. Distribution of average prediction errors over $\hat{r}_{j,t}$ percentiles.

TABLE III
LONG-SHORT SPREAD RETURNS*

Investment	Equal weights		Value weights		Trades per day
Model	Annual. return	Sharpe Ratio	Annual. return	Sharpe Ratio	
LSTM^{bin}	0.62	1.23	0.59	0.88	39.2
LSTM^{log}	0.64	1.25	0.53	0.80	32.7
LSTM^{mse}	0.57	1.06	0.50	0.78	39.8
MR	0.60	1.15	0.38	0.58	14.8
WR	0.59	1.16	0.52	0.79	26.2
MWR	0.25	0.56	0.09	0.13	40.5
Buy & Hold	0.13	0.41	0.13	0.49	–

*absolute returns after transaction costs on an initial investment of $20,000.

Our quality measure is accuracy, which determines the proportion of true classifications in all classifications. Table II reports the prediction accuracy of the models. All models correctly classify more than 50% of returns. The mean accuracy across all models is 52%, consistent with [9]. We also determine the cross-sectional correlation of signals and realized returns. Most signals have a correlation of about 4% with subsequent realized returns.

Finally, we employ the Diebold-Mariano test [31] and find empirical evidence against the null that the difference between predictions of our customized LSTMs and the benchmark models is zero (all p-values < 0.05). Even when adjusting the critical p-value to 0.00625 to control for multiple testing (Bonferroni correction), we reject the null in seven out of eight tests. We conclude that our proposed loss function provides investors with superior forecasts.

C. Investment case study

In our practice test, we discuss how beneficial the models are for private investors and whether the loss function actually delivers on its promise to protect investors from unexpected losses. We assume a private investor who uses an online broker to trade shares every day at market close and pays $0.50 per trade and $0.004 per share traded. With most popular online brokers, private investors can go both long and short in a matter of seconds at the same cost. The investor buys the 15 shares with the highest return predictions and shorts the 15 shares with the lowest predictions. The portfolio Buy & Hold buys (sells) shares when they are traded for the first (last) time.

Table III shows the average annualized returns and Sharpe Ratios after transaction costs on a start investment of $20,000. The last column reports the average number of trades per day. Replacing one portfolio constituent incurs two trades. The customized LSTMs contribute positively to the portfolio performance and produce advantageous Sharpe Ratios and returns. Compared to the binary LSTM, the logistic LSTM model benefits from less trading activities and produces a higher annual return. Similarly, the MSE model suffers from frequent trading and underperforms the other LSTM models. When excluding the impact of small caps, portfolio returns decrease. Consistent with [20], LSTM and reversal signals prefer small caps over large caps.

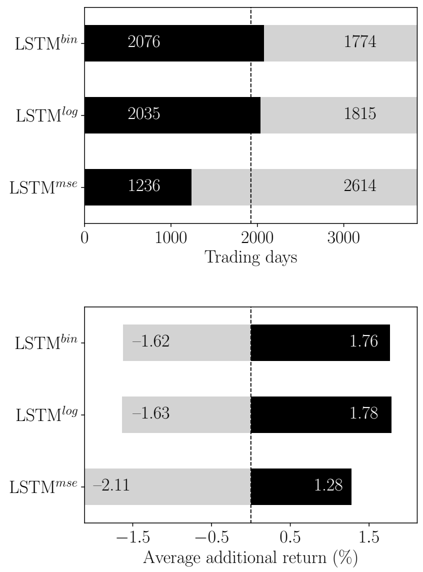

Fig. 3. Top: Trading days that generate unexpected profits (black) and losses (gray). Bottom: Average change in portfolio value for unexpected movements.

Next, we focus on the protection from unexpected portfolio losses. We designed the loss function with the idea of protecting investors from situations where the realized return is lower than the predicted return. The upper panel in Fig. 3 shows the fraction of days on which the investor faces unexpected losses (black) out of all days with prediction errors. The dashed line marks the half of all 3,850 trading days. While both customized LSTMs predict too large portfolio returns only on around 1,800 days, MSE investors experience unexpected losses on two-thirds of all trading days.

The bottom panel in Fig. 3 illustrates the average loss or profit for positive and negative prediction errors. In the case of a too low prediction, the customized models earn about 1.8% return in addition to the predicted return. For MSE investors, average unexpected losses exceed average unexpected gains. The mean unexpected profit in the binary model is $1.76\% \cdot 0.54 - 1.62\% \cdot 0.46 = 0.23\%$, and -1.03% in the MSE model. In other words, the MSE investor earns on average 1.03% less daily return than predicted, while the investor in the binary model earns 0.23% more daily return than predicted.

VII. LONG-HORIZON PREDICTIONS

There are two reasons why we should test our model for longer prediction and holding periods. First, the fractal markets hypothesis states that returns exhibit self-repeating patterns when viewed at different time scales [32]. Second, one may argue that our case study is not relevant for each private investor due to the frequent trading. We follow [9] in testing our models at weekly and monthly prediction scales and assume a private investor who uses the predictions to rebalance the portfolio once per week or month, respectively.

We consider share prices on the last day of a period (week or month) and calculate the returns achieved during that period. Accordingly, the LSTM processes only weekly

TABLE IV
LONG-SHORT SPREAD RETURNS* ON LONG-HORIZON PORTFOLIOS

Rebalancing	Every week		Every month	
Model	Annualized return	Sharpe Ratio	Annualized return	Sharpe Ratio
LSTM^{bin}	0.71	1.40	0.41	0.88
LSTM^{log}	0.73	1.54	0.66	1.20
LSTM^{mse}	0.65	1.24	0.65	1.16
WR (MR)	0.23	0.53	0.29	0.14

*absolute returns after transaction costs on an initial investment of $20,000.

(monthly) returns when predicting weekly (monthly) returns. We convert the sequence length of 60 days to 12 weeks in the weekly model and 3 months in the monthly model. The network configuration is identical to the daily model.

Table IV reports the average annualized returns and Sharpe Ratios after transaction costs. As prediction periods grow, LSTMs increase their lead over reversals. Returns on weekly revised LSTM portfolios triple returns on reversal portfolios. Compared to the daily model, lower transaction costs improve the performance, making weekly-revised LSTM portfolios superior to daily-revised portfolios.

VIII. REVIEW OF THE LOSS FUNCTION

A. Reversal imitation or unique learning?

One might wonder why our customized LSTMs outperform all other approaches. We try to shed light on this question by looking into how unique the signals of our customized LSTMs are. Fig. 4 illustrates the average preceding midweek, one-week and one-month returns measured for positive and negative return predictions. The upper (lower) part focuses on negative (positive) return forecasts. The large distance between the preceding returns of positive return predictions and the corresponding preceding returns of negative return predictions indicates imitation of reversal signals.

Next, we examine the relationships between portfolios using linear regression on portfolio returns and the market risk premium $r_m - r_f$. Portfolio returns are the equally weighted spreads between realized returns on buy signals and realized returns on sell signals. Table V shows the estimated parameters and the R^2 in the last row. The relationships of LSTMs with

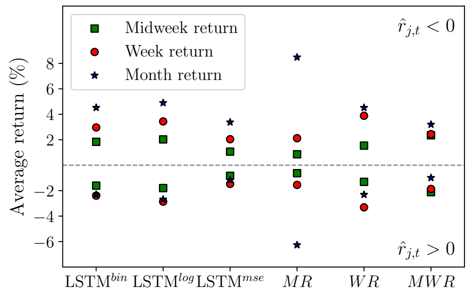

Fig. 4. Average preceding returns for negative (upper part) and positive (lower part) return forecasts $\hat{r}_{j,t}$.

TABLE V
LINEAR REGRESSION ON PORTFOLIO RETURNS

Model	LSTM^{bin}	LSTM^{log}	LSTM^{mse}
Alpha	0.02***	0.02***	0.01*
$r_m - r_f$	0.08***	0.04**	0.09***
MR	0.09***	0.09***	0.05***
WR	0.25***	0.26***	0.25***
MWR	0.32***	0.33***	0.17***
R^2	0.55	0.63	0.28

***, **, * indicate significance at 1%, 5%, 10% level.

TABLE VI
LINEAR REGRESSION ON REALIZED RETURNS

Predictor	R^2 (%)
Market data	1.97
Market data & LSTM^{mse}	1.99
Market data & LSTM^{bin}	2.24
Market data & LSTM^{log}	2.23

reversal portfolios are stronger than with the market portfolio. LSTMs extract reversal signals mainly over time horizons of one week and shorter. While the R^2 is larger than in [3] and [9], the association with reversal signals is consistent with [3]. Reversals explain slightly more than half of the variation in portfolio returns, but this in turn means that they also fail to explain slightly less than half of the variation in returns.

Lastly, we follow [9] and compare the R^2 of linear approximations to realized returns with and without controlling for return predictions. Table VI focuses on regressions of realized returns on the market predictors ($r_{j,t}$, $r_{m,t}$) in the first model, and on the market predictors and return predictions in the following models. While the MSE predictions do not increase the R^2, predictions produced by the customized LSTMs improve the determination by up to 14%. Although we document similarities with reversal signals, the loss functions provide unique information undiscovered by the standard models that lead to superior knowledge.

B. Loss functions in linear approximations to returns

LSTMs are often said to be superior compared to other time series models [25] because of their capability of capturing nonlinearity in the data. However, the question arises whether linear approximations to returns would also improve with our customized loss function. We perform a linear approximation to realized returns minimizing the $WMSE$ on the training and validation set and then use the estimated coefficients to predict returns for the test set.

Table VII shows the prediction quality and equally weighted long-short spread returns after transaction costs. We omit the binary loss function since the coefficients are identical with the MSE model, i.e., all losses are 'bad' losses and receive the same weights. Despite the prediction quality competing with LSTMs, the portfolios exhibit a high trading frequency and underperform the market, suggesting poor nominal return predictions at the distribution tails. The MSE function provides better results than the $WMSE$ function.

TABLE VII
PREDICTION QUALITY AND LONG-SHORT SPREAD RETURNS* OF LINEAR MODELS

Model	Accuracy (%)	Correlation (%)	Annualized return	Sharpe Ratio
LIN^{log}	51.85	3.71	0.10	0.23
LIN^{mse}	51.92	3.84	0.12	0.34

*absolute returns after transaction costs on an initial investment of $20,000.

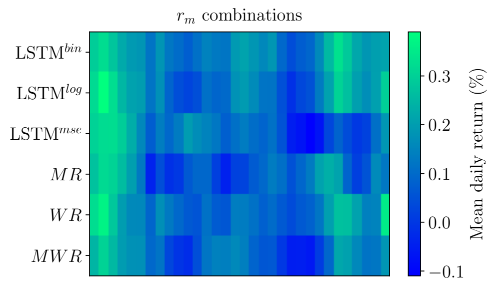

Fig. 5. Heatmap of long-short spread returns on the test data set.

IX. ROBUSTNESS CHECK: MARKET ANALYSIS

Patterns in financial time series captured by LSTMs may reflect market sentiment. One question that arises is whether our models perform equally well in different market environments or exhibit systematic differences. As a metric for the market environment, we develop 32 sentiment classes representing all possible combinations of signs of the market return r_m during the last five trading days prior to a prediction. The first (last) class represents five consecutive negative (positive) market returns, the second class represents four consecutive negative followed by a positive market return, and so on.

Fig. 5 plots the average daily long-short spread returns during the test period across all r_m combinations. From left to right, market sentiment changes from very positive (first class) to very negative (last class). The map emphasizes that each predictive model has its strengths in different situations. All models perform slightly better in the right and left quarters of the map, which group combinations with three consecutive positive and negative returns at the beginning of the week, respectively. Returns on the customized LSTMs are superior to the benchmark models in most combinations, indicating the robustness of the loss function to market sentiment.

X. CONCLUSION

In this paper, we present a novel approach that connects return prediction with the actual financial impact of prediction errors on portfolio performance. To this end, we introduce a customized loss function for return prediction that teaches LSTMs to underpredict rather than overpredict returns. We show that annualized returns on regularly revised portfolios increase by an average of five percentage points over the standard LSTM and reversal strategies.

One limitation of our approach, however, is that both the computing power required for the training and the technical infrastructure for automated trading with connected live market data are not easy to set up for private investors. Yet, our model does not require costly retraining: Once the model is trained and set up, it runs with very little effort.

Overall, our proposed loss function is beneficial in nonlinear optimization and extracts unique signals in a return-predicting LSTM model. Moreover, we find robust performance throughout a wide range of market sentiments.

REFERENCES

[1] R. Sullivan, A. Timmermann, and H. White, "Data-snooping, technical trading rule performance, and the bootstrap," *Journal of Finance*, vol. 54, no. 5, pp. 1647–1691, 1999.
[2] P. Bajgrowicz and O. Scaillet, "Technical trading revisited: False discoveries, persistence tests, and transaction costs," *Journal of Financial Economics*, vol. 106, no. 3, pp. 473–491, 2012.
[3] T. Fischer and C. Krauss, "Deep learning with long short-term memory networks for financial market predictions," *European Journal of Operational Research*, vol. 270, no. 2, pp. 654–669, 2018.
[4] L. Chen, M. Pelger, and J. Zhu, "Deep learning in asset pricing," *Available at SSRN 3350138*, 2020, unpublished.
[5] S. Borovkova and I. Tsiamas, "An ensemble of lstm neural networks for high-frequency stock market classification," *Journal of Forecasting*, vol. 38, no. 6, pp. 600–619, 2019.
[6] M. Fabbri and G. Moro, "Dow jones trading with deep learning: The unreasonable effectiveness of recurrent neural networks," in *Proceedings of the 7th International Conference on Data Science, Technology and Applications*, ser. DATA 2018. SciTePress, 2018, pp. 142—-153.
[7] A. Flori and D. Regoli, "Revealing pairs-trading opportunities with long short-term memory networks," *European Journal of Operational Research*, vol. 295, no. 2, pp. 772–791, 2021.
[8] S. Gu, B. T. Kelly, and D. Xiu, "Empirical asset pricing via machine learning," *The Review of Financial Studies*, vol. 33, no. 5, pp. 2223–2273, 02 2020.
[9] J. Jiang, B. T. Kelly, and D. Xiu, "(re-) imag (in) ing price trends," *Chicago Booth Research Paper*, vol. 21-01, 2020.
[10] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.
[11] E. F. Fama, "The behavior of stock-market prices," *Journal of Business*, vol. 38, no. 1, pp. 34–105, 1965.
[12] N. Jegadeesh, "Evidence of predictable behavior of security returns," *Journal of Finance*, vol. 45, no. 3, pp. 881–898, 1990.
[13] B. N. Lehmann, "Fads, martingales, and market efficiency," *The Quarterly Journal of Economics*, vol. 105, no. 1, pp. 1–28, 1990.
[14] N. Jegadeesh and S. Titman, "Returns to buying winners and selling losers: Implications for stock market efficiency," *Journal of Finance*, vol. 48, no. 1, pp. 65–91, 1993.
[15] ——, "Overreaction, delayed reaction, and contrarian profits," *Review of Financial Studies*, vol. 8, no. 4, pp. 973–993, 1995.
[16] A. Subrahmanyam, "Distinguishing between rationales for short-horizon predictability of stock returns," *Financial Review*, vol. 40, no. 1, pp. 11–35, 2005.
[17] J. Boudoukh, M. P. Richardson, and R. F. Whitelaw, "A tale of three schools: Insights on autocorrelations of short horizon stock returns," *Review of Financial Studies*, vol. 7, pp. 539–573, 1994.
[18] D. Avramov, T. Chordia, and A. Goyal, "Liquidity and autocorrelations in individual stock returns," *The Journal of Finance*, vol. 61, no. 5, pp. 2365–2394, 2006.
[19] E. F. Fama, "Efficient capital markets: A review of theory and empirical work," *Journal of Finance*, vol. 25, no. 2, pp. 383–417, 1970.
[20] D. Avramov, S. Cheng, and L. Metzker, "Machine learning versus economic restrictions: Evidence from stock return predictability," *Available at SSRN 3450322*, 2021, unpublished.
[21] K. Chen, Y. Zhou, and F. Dai, "A lstm-based method for stock returns prediction: A case study of china stock market," in *2015 IEEE International Conference on Big Data (Big Data)*, 2015, pp. 2823–2824.
[22] W. Bao, J. Yue, and Y. Rao, "A deep learning framework for financial time series using stacked autoencoders and long-short term memory," *PLoS ONE*, vol. 12, no. 7, 2017.

[23] C. Sang and M. Di Pierro, "Improving trading technical analysis with tensorflow long short-term memory (lstm) neural network," *Journal of Finance and Data Science*, vol. 5, no. 1, pp. 1–11, 2019.

[24] D. M. Q. Nelson, A. C. M. Pereira, and R. A. de Oliveira, "Stock market's price movement prediction with lstm neural networks," in *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017, pp. 1419–1426.

[25] S. Siami-Namini, N. Tavakoli, and A. Siami Namin, "A comparison of arima and lstm in forecasting time series," in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2018, pp. 1394–1401.

[26] L. Troiano, E. M. Villa, and V. Loia, "Replicating a trading strategy by means of lstm for financial industry applications," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 3226–3234, 2018.

[27] J. Qiu, B. Wang, and C. Zhou, "Forecasting stock prices with long-short term memory neural network based on attention mechanism," *PLOS ONE*, vol. 15, no. 1, pp. 1–15, 01 2020.

[28] Refinitiv, "Pricing and market data," www.refinitiv.com/en/financial-data/pricing-and-market-data, 2021.

[29] F. Chollet *et al.*, "Keras," www.keras.io, 2015.

[30] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: www.tensorflow.org/

[31] F. Diebold and R. Mariano, "Comparing predictive accuracy," *Journal of Business & Economic Statistics*, vol. 13, no. 3, pp. 253–63, 1995.

[32] B. Mandelbrot, P. Cootner, R. Gomory, E. Fama, W. Morris, and H. Taylor, *Fractals and Scaling in Finance: Discontinuity, Concentration, Risk. Selecta Volume E*. Springer Science & Business Media, New York, 2013.